

SAMS  
**Teach  
Yourself**

- 全球销量逾百万册的系列图书
- 连续十余年打造的经典品牌
- 直观、循序渐进的学习教程
- 掌握关键知识的最佳起点
- “Read Less, Do More”（精读多练）的教学理念
- 以示例引导读者完成最常见的任务

每章内容针对初学者精心设计，**1**小时轻松阅读学习，  
**24**小时彻底掌握关键知识

每章**案例与练习题**助你轻松完成常见任务，  
通过**实践**提高应用技能，巩固所学知识

随书附赠  
光盘

包含用于Windows  
Linux和Mac OS X  
的启动工具包

# PHP、MySQL 和 Apache

## 入门经典（第5版）

[美] Julie C. Meloni 著  
李军 译

人民邮电出版社  
POSTS & TELECOM PRESS

# 目 录

[版权信息](#)

[版权声明](#)

[内容提要](#)

[致 谢](#)

[作者简介](#)

[前 言](#)

[第1部分 基础知识](#)

[第1章 安装QuickStart向导](#)

[1.1 使用第三方的安装包](#)

[1.2 Linux/UNIX下的安装](#)

[1.3 在Windows上安装XAMPP](#)

[1.4 在Mac OS X上安装XAMPP](#)

[1.5 让XAMPP更安全](#)

[1.6 故障排除](#)

[第2章 安装和配置MySQL](#)

[2.1 MySQL的当前版本和未来版本](#)

[2.2 如何获取MySQL](#)

[2.3 在Linux/UNIX上安装MySQL](#)

[2.4 在Mac OS X上安装MySQL](#)

[2.5 在Windows上安装MySQL](#)

[2.6 安装故障排除](#)

[2.7 基本安全规则](#)

[2.7.1 启动MySQL](#)

[2.7.2 增强MySQL连接的安全](#)

[2.8 MySQL权限系统简介](#)



[2.8.1 两步身份验证过程](#)

[2.8.2 添加用户](#)

[2.8.3 移除权限](#)

[2.9 小结](#)

[2.10 Q&A](#)

[2.11 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

[第3章 安装和配置Apache](#)

[3.1 Apache的当前版本及未来版本](#)

[3.2 选择合适的安装方法](#)

[3.2.1 从源代码安装](#)

[3.2.2 安装一个二进制代码版本](#)

[3.3 在Linux/UNIX上安装Apache](#)

[3.3.1 下载Apache源代码](#)

[3.3.2 解压源代码](#)

[3.3.3 准备编译Apache](#)

[3.3.4 编译和安装Apache](#)

[3.4 在Mac OS X上安装Apache](#)

[3.5 在Windows上安装Apache](#)

[3.6 Apache配置文件结构](#)

[3.6.1 指令](#)

[3.6.2 容器](#)

[3.6.3 条件评估](#)

[3.6.4 ServerRoot指令](#)

[3.6.5 per-directory配置文件](#)

[3.7 Apache日志文件](#)

[3.7.1 access\\_log文件](#)

[3.7.2 error\\_log文件](#)

[3.7.3 其他文件](#)

[3.8 Apache相关命令](#)

[3.8.1 Apache服务器二进制程序](#)

[3.8.2 Apache控制脚本](#)

[3.9 第一次启动Apache](#)

[3.9.1 检查你的配置文件](#)

[3.9.2 启动Apache](#)  
[3.10 故障排除](#)  
[3.10.1 已有Web服务器](#)  
[3.10.2 不允许绑定到端口](#)  
[3.10.3 拒绝访问](#)  
[3.10.4 错误组设置](#)  
[3.11 小结](#)  
[3.12 Q&A](#)  
[3.13 实践练习](#)  
[练习题](#)  
[解答](#)  
[思考题](#)

[第4章 安装和配置PHP](#)  
[4.1 PHP的当前版本和未来版本](#)  
[4.2 在带有Apache的Linux/UNIX上编译PHP](#)  
[4.2.1 额外的Linux/UNIX配置选项](#)  
[4.2.2 在Linux/UNIX上集成PHP和Apache](#)  
[4.3 在Mac OS X上安装PHP](#)  
[4.4 在Windows上安装PHP](#)  
[在Windows上集成PHP和Apache](#)  
[4.5 php.ini基础](#)  
[4.6 测试安装](#)  
[4.7 获取安装帮助](#)  
[4.8 PHP脚本基础](#)  
[4.8.1 开始和结束一个PHP语句块](#)  
[4.8.2 echo语句和print\(\)函数](#)  
[4.8.3 组合HTML和PHP](#)  
[4.8.4 为PHP代码添加注释](#)  
[4.9 小结](#)  
[4.10 Q&A](#)  
[实践练习](#)  
[问答题](#)  
[解答](#)  
[思考题](#)

[第2部分 PHP语言结构](#)

## [第5章 PHP的组成部分](#)

### [5.1 变量](#)

#### [5.1.1 全局变量](#)

#### [5.1.2 超全局变量](#)

### [5.2 数据类型](#)

#### [5.2.1 使用settype\(\)来改变变量的数据类型](#)

#### [5.2.2 通过类型转换改变变量的数据类型](#)

#### [5.2.3 为何测试类型](#)

### [5.3 操作符和表达式](#)

#### [5.3.1 赋值操作符](#)

#### [5.3.2 算术操作符](#)

#### [5.3.3 连接操作符](#)

#### [5.3.4 复合赋值操作符](#)

#### [5.3.5 自动增加和减少一个整型变量](#)

#### [5.3.6 比较操作符](#)

#### [5.3.7 使用逻辑操作符创建复杂的测试表达式](#)

#### [5.3.8 操作符优先级](#)

### [5.4 常量](#)

### [5.5 小结](#)

### [5.6 Q&A](#)

### [5.7 实践练习](#)

#### [问答题](#)

#### [解答](#)

#### [思考题](#)

## [第6章 PHP的流程控制功能](#)

### [6.1 转换流程](#)

#### [6.1.1 if语句](#)

#### [6.1.2 使用else子句的if语句](#)

#### [6.1.3 使用带有elseif子句的if语句](#)

#### [6.1.4 switch语句](#)

#### [6.1.5 使用? 运算符](#)

### [6.2 循环](#)

#### [6.2.1 while语句](#)

#### [6.2.2 do...while语句](#)

#### [6.2.3 for语句](#)

#### [6.2.4 用break语句跳出循环](#)



[6.2.5 用continue语句跳过迭代](#)

[6.2.6 嵌套循环](#)

[6.3 代码块和浏览器输出](#)

[6.4 小结](#)

[6.5 Q&A](#)

[6.6 实践练习](#)

[练习题](#)

[解答](#)

[思考题](#)

[第7章 使用函数](#)

[7.1 什么是函数](#)

[7.2 调用函数](#)

[7.3 定义一个函数](#)

[7.4 从用户定义的函数返回值](#)

[7.5 变量作用域](#)

[使用global语句访问变量](#)

[7.6 使用static语句在函数调用之间保存状态](#)

[7.7 关于参数的更多内容](#)

[7.7.1 为参数设置默认值](#)

[7.7.2 把变量引用传递给函数](#)

[7.8 测试函数是否存在](#)

[7.9 小结](#)

[7.10 Q&A](#)

[7.11 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

[第8章 使用数组](#)

[8.1 什么是数组](#)

[8.2 创建数组](#)

[8.2.1 创建关联数组](#)

[8.2.2 创建多维数组](#)

[8.3 一些和数组相关的函数](#)

[8.4 小结](#)

[8.5 Q&A](#)

## [8.6 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

## [第9章 使用对象](#)

### [9.1 创建一个对象](#)

#### [9.1.1 对象的属性](#)

#### [9.1.2 对象方法](#)

#### [9.1.3 构造方法](#)

### [9.2 对象继承](#)

### [9.3 小结](#)

### [9.4 Q&A](#)

### [9.5 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

## [第3部分 深入编程](#)

## [第10章 使用字符串、日期和时间](#)

### [10.1 使用PHP格式化字符串](#)

#### [10.1.1 使用printf\(\)](#)

#### [10.1.2 指定一个字段宽度](#)

#### [10.1.3 参数交换](#)

#### [10.1.4 存储一个格式化字符串](#)

### [10.2 了解PHP中的字符串](#)

#### [10.2.1 索引字符串的一个注意事项](#)

#### [10.2.2 使用strlen\(\)获取一个字符串的长度](#)

#### [10.2.3 使用strstr\(\)获取一个字符串的子串](#)

#### [10.2.4 使用strpos\(\)找到一个子字符串的位置](#)

#### [10.2.5 使用substr\(\)提取一个字符串的一部分](#)

#### [10.2.6 使用strtok\(\)分解一个字符串](#)

### [10.3 在PHP中操作字符串](#)

#### [10.3.1 使用trim\(\)、ltrim\(\)和strip\\_tags\(\)整理一个字符串](#)

#### [10.3.2 使用substr\\_replace\(\)替换一个字符串的一部分](#)

#### [10.3.3 使用str\\_replace\(\)替换子字符串](#)

- [10.3.4 转换大小写](#)
- [10.3.5 使用wordwrap\(\)和nl2br\(\)换行文本](#)
- [10.3.6 使用explode\(\)把字符串分解到数组](#)
- [10.4 使用PHP中的日期和时间函数](#)
  - [10.4.1 使用time\(\)获取日期](#)
  - [10.4.2 使用getdate\(\)转换一个时间戳](#)
  - [10.4.3 使用date\(\)转换一个时间戳](#)
  - [10.4.4 使用mktime\(\)创建时间戳](#)
  - [10.4.5 使用checkdate\(\)测试日期](#)
- [10.5 其他字符串、日期和时间函数](#)
- [10.6 小结](#)
- [10.7 实践练习](#)
- [10.8 Q&A](#)
  - [问答题](#)
  - [解答](#)
  - [思考题](#)

- [第11章 使用表单](#)
  - [11.1 创建一个简单的输入表单](#)
  - [11.2 使用用户定义数组访问表单输入](#)
  - [11.3 在单个页面上组合HTML和PHP代码](#)
  - [11.4 使用隐藏字段来保存状态](#)
  - [11.5 重定向用户](#)
  - [11.6 根据表单提交发送邮件](#)
    - [11.6.1 mail\(\)函数的系统配置](#)
    - [11.6.2 创建表单](#)
    - [11.6.3 创建发送邮件的脚本](#)
    - [11.6.4 使用HTML格式化邮件](#)
  - [11.7 使用文件上传](#)
    - [11.7.1 创建文件上传表单](#)
    - [11.7.2 创建一个文件上传脚本](#)
  - [11.8 小结](#)
  - [11.9 Q&A](#)
  - [11.10 实践练习](#)
    - [问答题](#)
    - [解答](#)
    - [思考题](#)



## [第12章 使用Cookie和用户会话](#)

### [12.1 Cookie简介](#)

#### [12.1.1 深入了解一个cookie](#)

#### [12.1.2 访问cookies](#)

### [12.2 使用PHP设置一个cookie](#)

#### [删除一个cookie](#)

### [12.3 会话函数概览](#)

### [12.4 开始一个会话](#)

### [12.5 使用会话变量](#)

### [12.6 销毁会话和重置变量](#)

### [12.7 在一个带有注册用户的环境中使用会话](#)

#### [12.7.1 使用注册的用户](#)

#### [12.7.2 使用用户偏好](#)

### [12.8 小结](#)

### [12.9 Q&A](#)

### [12.10 实践练习](#)

#### [问答题](#)

#### [解答](#)

#### [思考题](#)

## [第13章 使用文件和目录](#)

### [13.1 使用include语句包含文件](#)

#### [13.1.1 从一个被包含文档返回一个值](#)

#### [13.1.2 在控制结构中使用include语句](#)

#### [13.1.3 使用include\\_once语句](#)

#### [13.1.4 include\\_path命令](#)

### [13.2 验证文件](#)

#### [13.2.1 使用file\\_exists\(\)检查文件的存在性](#)

#### [13.2.2 文件还是目录](#)

#### [13.2.3 检查一个文件的状态](#)

#### [13.2.4 使用filesize\(\)确定文件的大小](#)

#### [13.2.5 获取有关一个文件的日期信息](#)

#### [13.2.6 编写一个执行多文件测试的函数](#)

### [13.3 创建并删除文件](#)

### [13.4 打开一个文件供写入、读取或添加](#)

### [13.5 读取文件](#)

#### [13.5.1 使用fgets\(\)和feof\(\)从一个文件读取行](#)

[13.5.2 使用fread\(\)函数从文件读取任意数量的数据](#)

[13.5.3 使用fgetc\(\)从文件读取字符](#)

[13.5.4 用file\\_get\\_contents\(\)读取文件内容](#)

[13.6 写入文件或向文件添加内容](#)

[13.6.1 使用fwrite\(\)或fputs\(\)写入文件](#)

[13.6.2 使用file\\_put\\_contents\(\)写文件内容](#)

[13.6.3 使用flock\(\)锁定文件](#)

[13.7 使用目录](#)

[13.7.1 使用mkdir\(\)创建目录](#)

[13.7.2 使用rmdir\(\)删除一个目录](#)

[13.7.3 使用opendir\(\)打开一个目录以供读取](#)

[13.7.4 使用readdir\(\)从一个目录读取内容](#)

[13.8 使用popen\(\)打开到进程和离开进程的管道](#)

[13.9 使用exec\(\)运行命令](#)

[13.10 使用system\(\)或passthru\(\)运行命令](#)

[13.11 小结](#)

[13.12 Q&A](#)

[13.13 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

[第14章 使用图像](#)

[14.1 理解图像创建过程](#)

[关于颜色](#)

[14.2 对PHP的必要修改](#)

[14.3 绘制一个新的图像](#)

[14.3.1 绘制形状和线条](#)

[14.3.2 使用颜色填充](#)

[14.4 绘制有趣的饼图](#)

[14.5 修改已有图像](#)

[14.6 使用来自用户输入的图像创建图像](#)

[14.7 使用脚本创建的图像](#)

[14.8 小结](#)

[14.9 Q&A](#)

[14.10 实践练习](#)

[问答题](#)

[解答](#)  
[思考题](#)

## [第4部分 PHP与MySQL整合](#)

### [第15章 理解数据库设计过程](#)

#### [15.1 良好的数据库设计的重要性](#)

#### [15.2 表关系的类型](#)

##### [15.2.1 一对一关系](#)

##### [15.2.2 一对多关系](#)

##### [15.2.3 多对多关系](#)

#### [15.3 理解规范化](#)

##### [15.3.1 平表带来的问题](#)

##### [15.3.2 第一范式](#)

##### [15.3.3 第二范式](#)

##### [15.3.4 第三范式](#)

#### [15.4 遵从设计过程](#)

#### [15.5 小结](#)

#### [15.6 Q&A](#)

#### [15.7 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

### [第16章 SQL基本命令](#)

#### [16.1 MySQL数据类型](#)

##### [16.1.1 数字数据类型](#)

##### [16.1.2 日期和时间类型](#)

##### [16.1.3 字符串类型](#)

#### [16.2 表的创建语法](#)

#### [16.3 使用INSERT命令](#)

#### [进一步学习INSERT语句](#)

#### [16.4 使用SELECT命令](#)

##### [16.4.1 排序SELECT结果](#)

##### [16.4.2 限制结果](#)

#### [16.5 在查询中使用WHERE](#)

##### [16.5.1 在WHERE子句中使用操作符](#)



- [16.5.2 使用LIKE比较字符串](#)
- [16.6 从多个表中查询](#)
  - [16.6.1 使用JOIN](#)
  - [16.6.2 使用子查询](#)
- [16.7 使用UPDATE命令来修改记录](#)
  - [16.7.1 条件式UPDATE](#)
  - [16.7.2 在UPDATE中使用已有的列值](#)
- [16.8 使用REPLACE命令](#)
- [16.9 使用DELETE命令](#)
  - [条件式DELETE](#)
- [16.10 MySQL中常用的字符串函数](#)
  - [16.10.1 长度和连接函数](#)
  - [16.10.2 截断和填充函数](#)
  - [16.10.3 定位和位置函数](#)
  - [16.10.4 子字符串函数](#)
  - [16.10.5 字符串修改函数](#)
- [16.11 在MySQL中使用日期和时间函数](#)
  - [16.11.1 操作日期](#)
  - [16.11.2 操作月份和年份](#)
  - [16.11.3 操作周](#)
  - [16.11.4 操作小时、分钟和秒](#)
  - [16.11.5 使用MySQL格式化日期和时间](#)
  - [16.11.6 使用MySQL执行日期算术](#)
  - [16.11.7 特殊函数和转换函数](#)
- [16.12 小结](#)
- [16.13 Q&A](#)
- [16.14 实践练习](#)
  - [练习题](#)
  - [解答](#)
  - [思考题](#)

## [第17章 使用MySQL中的事务和存储过程](#)

- [17.1 什么是事务](#)
  - [17.1.1 事务中使用的基本语法](#)
  - [17.1.2 使用事务的例子](#)
- [17.2 什么是存储过程](#)
- [17.3 小结](#)

[17.4 Q&A](#)

[17.5 实践练习](#)

[练习题](#)

[解答](#)

[思考题](#)

[第18章 使用PHP和MySQL交互](#)

[18.1 MySQL函数和MySQLi函数](#)

[18.2 使用PHP连接MySQL](#)

[18.2.1 进行连接](#)

[18.2.2 执行查询](#)

[18.2.3 获取错误消息](#)

[18.3 使用MySQL数据](#)

[18.3.1 避免SQL注入](#)

[18.3.2 使用PHP插入数据](#)

[18.3.3 使用PHP获取数据](#)

[18.3.4 PHP中其他的MySQL函数](#)

[18.4 小结](#)

[18.5 Q&A](#)

[18.6 实践练习](#)

[练习题](#)

[解答](#)

[思考题](#)

[第5部分 基本项目](#)

[第19章 管理一个简单的邮件列表](#)

[19.1 开发订阅机制](#)

[19.1.1 创建subscribers表](#)

[19.1.2 为共同函数创建一个包含文件](#)

[19.1.3 创建订阅表单](#)

[19.2 开发邮件发送机制](#)

[19.3 小结](#)

[19.4 Q&A](#)

[19.5 实践练习](#)

[问答题](#)

[解答](#)

## [思考题](#)

### [第20章 创建一个在线地址簿](#)

#### [20.1 规划和创建数据库表](#)

#### [20.2 为共同函数创建一个包含文件](#)

#### [20.3 创建一个菜单](#)

#### [20.4 创建记录添加机制](#)

#### [20.5 浏览记录](#)

#### [20.6 创建记录的删除机制](#)

#### [20.7 为一条记录添加子条目](#)

#### [20.8 小结](#)

#### [20.9 Q&A](#)

#### [20.10 实践练习](#)

#### [问答题](#)

#### [解答](#)

#### [思考题](#)

### [第21章 创建一个简单的讨论论坛](#)

#### [21.1 设计数据库表](#)

#### [21.2 为共同函数创建一个包含文件](#)

#### [21.3 创建输入表单和脚本](#)

#### [21.4 显示主题列表](#)

#### [21.5 显示一个主题中的帖子](#)

#### [21.6 向主题添加帖子](#)

#### [21.7 小结](#)

#### [21.8 Q&A](#)

#### [21.9 实践练习](#)

#### [问答题](#)

#### [解答](#)

#### [思考题](#)

### [第22章 创建一个在线商店](#)

#### [22.1 规划和创建数据库表](#)

##### [22.1.1 向store\\_categories表插入记录](#)

##### [22.1.2 向store\\_items表插入记录](#)

##### [22.1.3 向store\\_item\\_size表中插入记录](#)

##### [22.1.4 向store\\_item\\_color表插入记录](#)



[22.2 显示商品分类](#)

[22.3 显示商品](#)

[22.4 小结](#)

[22.5 Q&A](#)

[22.6 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

[第23章 创建一个购物车机制](#)

[23.1 规划和创建数据库表](#)

[23.2 把购物车整合到商店](#)

[23.2.1 把项目添加到购物车](#)

[23.2.2 浏览购物车](#)

[23.2.3 从购物车中删除项目](#)

[23.3 支付方法和结账过程](#)

[23.3.1 创建结账页面](#)

[23.3.2 执行结账操作](#)

[23.4 小结](#)

[23.5 Q&A](#)

[23.6 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

[第24章 创建一个简单的日历](#)

[24.1 构建一个简单的显示日历](#)

[24.1.1 检查用户输入](#)

[24.1.2 构建HTML表单](#)

[24.1.3 创建日历表格](#)

[24.1.4 向日历添加事件](#)

[24.2 创建一个日历库](#)

[24.3 小结](#)

[24.4 Q&A](#)

[24.5 实践练习](#)

[问答题](#)

[解答](#)

## [思考题](#)

### [第25章 限制对应用程序的访问](#)

#### [25.1 验证概览](#)

##### [25.1.1 客户机验证](#)

##### [25.1.2 用户管理方法](#)

#### [25.2 Apache验证模块功能](#)

##### [25.2.1 基于文件的验证](#)

##### [25.2.2 基于数据库文件的访问控制](#)

#### [25.3 使用Apache进行访问控制](#)

##### [25.3.1 实现访问规则](#)

##### [25.3.2 应用访问规则](#)

#### [25.4 组合Apache访问方法](#)

#### [25.5 根据HTTP方法限制访问](#)

#### [25.6 根据cookie值限制访问](#)

##### [25.6.1 创建授权用户表](#)

##### [25.6.2 创建登录表单和脚本](#)

##### [25.6.3 测试auth cookie](#)

#### [25.7 小结](#)

#### [25.8 Q&A](#)

#### [25.9 实践练习](#)

## [问答题](#)

## [解答](#)

## [思考题](#)

### [第26章 记录并监视Web服务器活动](#)

#### [26.1 标准Apache访问日志](#)

##### [26.1.1 确定记录什么](#)

##### [26.1.2 记录对文件的访问](#)

##### [26.1.3 记录对一个程序的访问](#)

#### [26.2 标准Apache错误日志](#)

##### [26.2.1 把错误记录到一个文件](#)

##### [26.2.2 把错误记录到一个程序](#)

##### [26.2.3 syslog守护进程参数](#)

##### [26.2.4 LogLevel指令](#)

#### [26.3 管理Apache日志](#)

##### [26.3.1 解析主机名](#)

- [26.3.2 日志备份](#)
- [26.3.3 日志分析](#)
- [26.3.4 监视错误日志](#)
- [26.4 把自定义信息记录到一个数据库](#)
- [26.4.1 创建数据库表](#)
- [26.4.2 创建PHP代码段](#)
- [26.4.3 创建示例报表](#)
- [26.5 小结](#)
- [26.6 Q&A](#)
- [26.7 实践练习](#)
- [问答题](#)
- [解答](#)
- [思考题](#)

- [第27章 应用程序本地化](#)
- [27.1 关于国际化和本地化](#)
- [27.2 关于字符集](#)
- [27.3 环境修改](#)
- [27.3.1 Apache的配置修改](#)
- [27.3.2 PHP的配置修改](#)
- [27.3.3 MySQL的配置修改](#)
- [27.4 创建一个本地化页面结构](#)
- [27.5 使用gettext\(\)来本地化应用程序](#)
- [27.6 小结](#)
- [27.7 Q&A](#)
- [27.8 实践练习](#)
- [问答题](#)
- [解答](#)
- [思考题](#)

- [第28章 使用XML](#)
- [28.1 什么是XML](#)
- [28.1.1 基本XML文档结构](#)
- [28.1.2 何时应该使用XML和PHP](#)
- [28.2 使用DOM函数在PHP中访问XML](#)
- [28.3 使用SimpleXML函数在PHP中访问XML](#)
- [28.4 使用JSON](#)

[28.5 小结](#)

[28.6 Q&A](#)

[28.7 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

## [第6部分 管理和优化](#)

### [第29章 Apache性能调校和虚拟主机](#)

[29.1 可扩展性问题](#)

[29.1.1 操作系统限制](#)

[29.1.2 和性能相关的Apache设置](#)

[29.2 使用ApacheBench载入测试](#)

[29.3 预先性能调校](#)

[29.3.1 把文件映射到内存](#)

[29.3.2 分布负载](#)

[29.3.3 缓存](#)

[29.3.4 减少数据传输](#)

[29.3.5 网络设置](#)

[29.4 防止滥用](#)

[29.5 实现虚拟主机](#)

[29.5.1 基于IP的虚拟主机](#)

[29.5.2 基于名字的虚拟主机](#)

[29.5.3 大量虚拟主机](#)

[29.6 小结](#)

[29.7 Q&A](#)

[29.8 实践练习](#)

[问答题](#)

[解答](#)

### [第30章 建立一个安全的Web服务器](#)

[30.1 安全性的需求](#)

[30.2 SSL协议](#)

[30.2.1 解决保密性需求](#)

[30.2.2 解决完整性的需求](#)

[30.2.3 解决验证的需求](#)

## [30.3 获取和安装SSL工具](#)

### [30.3.1 OpenSSL](#)

### [30.3.2 Apache的mod\\_ssl模块](#)

## [30.4 管理证书](#)

### [30.4.1 创建一个密钥对](#)

### [30.4.2 创建一个证书签发请求](#)

### [30.4.3 创建一个自签发的证书](#)

## [30.5 SSL配置](#)

### [启动服务器](#)

## [30.6 小结](#)

## [30.7 Q&A](#)

## [30.8 实践练习](#)

### [问答题](#)

### [解答](#)

## [第31章 优化和调校MySQL](#)

### [31.1 构建一个优化的平台](#)

#### [使用benchmark\(\)函数](#)

### [31.2 MySQL启动选项](#)

#### [关键启动参数](#)

### [31.3 优化表结构](#)

### [31.4 优化你的查询](#)

### [31.5 使用FLUSH命令](#)

### [31.6 使用SHOW命令](#)

#### [31.6.1 获取有关数据库和表的信息](#)

#### [31.6.2 获取表结构信息](#)

#### [31.6.3 获取系统状态](#)

### [31.7 小结](#)

## [31.8 Q&A](#)

## [31.9 实践练习](#)

### [问答题](#)

### [解答](#)

### [思考题](#)

## [第32章 软件升级](#)

### [32.1 停留在循环中](#)

#### [何时升级](#)

[32.2 升级MySQL](#)

[32.3 升级 Apache](#)

[修改Apache而不需要升级](#)

[32.4 升级PHP](#)

[使用PECL和PEAR扩展PHP](#)

[32.5 小结](#)

[32.6 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

[第33章 使用应用程序框架](#)

[33.1 理解应用程序框架](#)

[33.2 使用MVC模式](#)

[33.3 安装和使用PHP应用程序框架](#)

[33.3.1 Zend Framework](#)

[33.3.2 CakePHP](#)

[33.3.3 CodeIgniter](#)

[33.4 小结](#)

[33.5 实践练习](#)

[问答题](#)

[解答](#)

[思考题](#)

## 版权信息

书名：PHP、MySQL和Apache入门经典（第5版）

ISBN：978-7-115-30270-0

本书由人民邮电出版社发行数字版。版权所有，侵权必究。

---

您购买的人民邮电出版社电子书仅供您个人使用，未经授权，不得以任何方式复制和传播本书内容。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

---

• 著 [美] Julie C.Meloni

译 李 军

责任编辑 陈冀康

• 人民邮电出版社出版发行 北京市丰台区成寿寺路11号

邮编 100164 电子邮件 315@ptpress.com.cn



网址 <http://www.ptpress.com.cn>

- 读者服务热线: (010)81055410

反盗版热线: (010)81055315

## 版权声明

Julie C. Meloni: Sams Teach Yourself PHP, MySQL and Apache All in One

ISBN: 978-0-672-33543-3

Copyright © 2012 by Pearson Education, Inc.

Authorized translation from the English languages edition published by Pearson Education, Inc.

All rights reserved.

本书中文简体字版由美国**Pearson**公司授权人民邮电出版社出版。未经出版者书面许可，对本书任何部分不得以任何方式复制或抄袭。

版权所有，侵权必究。

## 内容提要

PHP、MySQL和Apache是Web应用开发的强大组合工具。本书针对这三种主流工具的最新版本，逐步介绍了如何安装、配置和使用这些工具组合，并通过一些典型的项目案例，帮助读者开发出功能强大的Web应用。

全书分为6个部分共33章。第1部分“基础知识”，包括第1章到第4章，引领读者深入了解、安装和配置MySQL、Apache和PHP。第2部分“PHP语言结构”，包括第5章到第9章，讲解PHP语言基础，包括数组和对象这样的结构化元素。第3部分“深入编程”，包括第10章到第14章，介绍中级应用程序开发的主题，包括使用表单和文件、限制访问以及完成包含某个专门概念的小项目。第4部分“PHP和MySQL整合”，包括第15章到第18章，介绍使用数据库的一般方法。第5部分“基本项目”，包括第19章到第28章，介绍如何整合前面已经学习到的所有知识，使用PHP和MySQL执行一个特定任务。第6部分“管理和优化”，包括第29章到第33章，介绍管理和优化Apache和MySQL的方法。

本书内容全面、讲解详细、由浅入深、实例丰富，而且考虑到读者使用不同操作系统和开发环境的需求。本书可作为PHP、MySQL、Apache初学者的学习指南，也可作为Web开发技术人员的参考用书。

## 致 谢

感谢Apache Software Foundation、PHP Group和MySQL AB，它们所创建的这些超级产品驱动了大量的Web，由此，它们得到了更多的认可。

尽管本书自从Daniel Lopez（*Sams Teach Yourself Apache 2 in 24 Hours* 的作者）和Matt Zandstra（*Sams Teach Yourself PHP in 24 Hours* 的作者）最初出版的版本之后已经经历了好几个版本，但是，如果没有他们多年前的工作的话，本书也不可能存在。

## 作者简介

Julie C. Meloni是一位技术顾问。Julie自从Web诞生的时候就开始从事基于Web应用的开发。她是多本关于Web编程语言和数据库的图书和文章的作者，包括Sams Teach Yourself Blogging in a Snap一书。她的博客是thickbook.com和nerdtripping.com，后者包含了很多关于旅游的提示和技巧。

## 前言

欢迎阅读本书。我很高兴地告诉你，PHP语言及其开发者社群和用户每天都在持续增加，因此，本书需要更新版本。

本书前一个版本介绍的PHP4，其“生命尽头”终于快到了，在GoPHP5活动的帮助下，Web主机服务提供商和应用程序开发者迁移其服务和代码，逐渐放弃特定于PHP 4的功能和代码实践，而进入PHP 5的世界。这是一个更快速、更好的功能集合。和本书上一版一样，本版中所有的代码都是基于编写本书的时候PHP最新的可用版本（具体来说就是PHP 5.4.0）。

你可能已经听说了PHP 6或者已经看到一些图书宣传自己使用了PHP 6作为核心语言。然而，PHP 6的语言版本还没有具体化，PHP 6计划的功能，都已添加到了PHP 5.3和PHP 5.4中。因此，如果你听说了PHP 6而在网上或PHP.net站点没有找到任何相关内容的话，不必担心，你并没有错过什么。

在本书中，你将学到配置和管理Apache Web服务器所必需的概念、PHP编程基础，以及使用和管理MySQL关系型数据库系统的方法。本书的目标是提供理解如何无缝地整合这些技术的基础，并且教授将它们整合到功能完备的Web站点和Web应用程序中的实用知识。本书应该是迈向高级站点开发的第一步，而不是唯一的一步。

## 本书的目标读者

本书为那些对基于Web的开发环境（可能是在Linux/UNIX或Windows下，甚至是在Mac OS X下）具有一般性理解的人们量身打造。假设你已经熟悉了自己的操作系统，并且掌握了编译（在Linux/UNIX系统上）或安装（在Windows和Mac OS X系统上）软件的基本方法。

假设读者没有任何关于语言的知识，这样的读者可先阅读介绍使用PHP编程的章节。然而，如果你有使用其他编程语言的经验，例如ASP、JSP、Ruby或Perl，你会发现这些章节学起来很容易，因为你已经熟悉诸如变量、控制结构、函数、对象等编程元素。类似的，如果你已经使用过其他的数据库，例如Oracle或Microsoft SQL Server，那会为学习和MySQL相关的内容奠定一个坚实的基础。

唯一真正需要的是，你能够理解使用HTML创建静态Web内容。如果你只是刚刚开始Web开发，应该能够使用本书，但是，你应该先考虑阅读一个HTML教程。如果你熟悉了创建基本的页面，那么你会学得更好。

## 本书的组织结构

本书分为6部分，对应特定的主题。你应该按部就班地学习每个部分，因为每个部分的内容都构建在前面部分的基础之上。

- 第1部分“基础知识”，提供一个安装的快速指导，并带领读者深入了解安装和配置MySQL、Apache和PHP的过程。在继续学习之前，你至少需要完成这些课程中的一种，要么快速安装，要么是更长的详细安装过程，除非你已经成功地安装了这些软件。即便你不需要在自己的环境中安装或配置MySQL、Apache和PHP，还是应该浏览一下这些课程，以便理解它们相互交互的基础。
- 第2部分“PHP语言结构”，教授PHP语言基础，包括数组和对象这样的结构化元素。那些示例将帮助你习惯编写代码，将这些示例上传到你的服务器，并测试其运行结果。
- 第3部分“深入编程”，介绍了中级应用程序开发的主题，包括使用表单和文件、限制访问以及完成一些小项目，这些小项目是设计用来介绍一个专门概念的。
- 第4部分“PHP和MySQL整合”，介绍使用数据库的一般方法，例如，数据库规范化，以及使用PHP来连接并操作MySQL。其中包含SQL基础知识，还包括特定于MySQL的函数和其他信息。
- 第5部分“基本项目”，介绍如何整合前面已经学习到的所有知识，使用PHP和MySQL执行一个特定任务。这些项目包括地址簿、一个讨论论坛和一个基本的网上商店。这些例子都是在一个黑白环境下构建的，就是说在美观性上显得很简约。这使你可以把精力集中在



程序设计和搭建结构所需的逻辑上，而不是在显示美观上。

- 第6部分“管理和优化”，介绍管理和优化Apache和MySQL。它还包含了有关虚拟主机以及建立一个安全Web服务器的信息。

如果你发现自己已经熟悉某个主题，可以跳过并继续向前学习。然而，某些地方会引用前面的章节中学习过的特定概念，因此，请注意必须浏览一下跳过的章节，以便保证你的开发环境和本书一致。

在每章的末尾，都有一些问答题来测试你对该章内容的掌握程度。附加的思考题则提供了应用该章知识的另外一种方式，并且引导你在下一章使用这些刚刚学习到的知识。

## 本书源代码

在各章中出现的程序清单中的所有代码，都可以在随书光盘中找到。也可以从作者的Web站点<http://www.thickbook.com/> 下载打包的代码。

自己录入代码，在打字、产生错误以及执行叫人伤透脑筋的查找分号错误的任务等方面会有些有用的体验。然而，如果你想要略过这些课程并且只是把本书的工作代码上传到你的站点，也没问题。

## 本书体例

本书使用不同的字体来表示代码和正文，也通过这种方法来帮你识别重要的概念。在本书中，代码、命令和你所输入的或者在计算机屏幕上看到的文本，都使用等宽字体。在正文中定义新术语的地方使用斜体。此外，特别的内容版块都带有图标。

- “提示”给出了和当前话题相关的一段有趣的信息。
- “你知道吗”提出建议，或者教给你执行一项任务。
- “注意”警告你潜在的缺陷并说明如何避免它们。

## 第1部分 基础知识

第1章 安装**QuickStart**向导

第2章 安装和配置**MySQL**

第3章 安装和配置**Apache**

第4章 安装和配置**PHP**

## 第1章 安装QuickStart向导

为了帮助你快速起步，这个简单的第1章将帮助你熟悉整体的跨平台安装软件包XAMPP的安装过程。后续的第2、3、4章分别介绍了如何从互联网上获取并安装MySQL、Apache和PHP，从而可以确保软件版本是最新的。另外，这几章还展开说明了安装过程中的每一步，以及理解这些技术如何一起工作的其他重要信息。

你应该在接下来的三章中熟悉每一种技术的扩展信息。然而，如果你只是想要开始在本地上工作的话，本章也是很好的参考。

## 1.1 使用第三方的安装包

第三方安装包是由最初创建者以外的公司或组织所提供的程序包。在本章中，我们将学习如何使用XAMPP安装包来同时安装PHP、MySQL和Apache，可以在我们将来使用的任何操作系统上（Linux/UNIX、Windows或Mac）完成安装。

除了因为我自己使用XAMPP数年了，我选择在本章中使用它的另一个原因是，其名称中带有X。X表示这是AMPP（Apache、MySQL、PHP和Perl）的一个跨平台安装程序（Perl不是本书的主题，因此，将其当做额外的学习内容）。还有两种其他很好的Apache、MySQL和PHP安装程序包，但它们针对特定的操作系统。

- **WAMP** ——在Windows上安装Apache、MySQL和PHP。参见 <http://www.wampserver.com/> for more information。
- **MAMP** ——在Mac上安装Apache、MySQL和PHP。参见 <http://www.mamp.info/> for more information。

使用第三方安装程序包的一个潜在的缺点是，绑定在一起的核心技术的版本，总是正式版之后的几个修订版。这恰好是因为创建和测试程序包本身涉及到一些工作，以便确保这些技术的最新版本之间没有冲突；这还必须经过一个质量保证的过程。然而，这一方案的优点是，当你使用安装程序包来安装这些技术的时候，升级的过程只不过是运行一个新的安装程序，它会负责为你删除或升级所有的文件。

接下来的3节介绍了XAMPP的基本安装过程。你只需要阅读适用于

你的操作系统的节。然而，一定要阅读本章1.5节，因为它适用于所有的操作系统。

## 1.2 Linux/UNIX下的安装

尽管下面介绍的过程是在Ubuntu Linux系统上测试过的，但这些步骤对其他所有Linux或商业UNIX发布的默认安装都是一样的。在编译过程中，你可能遇到意外的错误信息，此时应联系系统管理员或参考自己特定的操作系统的文档。

如果你使用本书随书光盘中包含的XAMPP版本，请从这里开始，以超级用户启动（作为root登录或作为一个常规系统用户su登录），并在文件系统中用/mnt参数加载CD-ROM。

```
# mount /dev/cdrom /mnt -t iso9660
```

现在，你已经访问了随书光盘中的XAMPP文件，或者你已经从<http://www.apachefriends.org/en/xampp-linux.html> 下载了最新版本，继续具体的安装步骤如下。

作为一名超级用户，将文件从随书光盘的XAMPP目录（或者从下载位置）复制到/opt目录。在/opt目录中，解压缩已经下载的文件如下。

```
# tar xvfz xampp-linux-VERSION-NUMBER.tar.gz -C /opt
```

### 注意：

在编写本书时，XAMPP的版本号是1.8.0-beta2，因此，其文件名是xampp-linux-1.8.0-beta2.tar.gz。随后的版本将具有不同的文件名，因此，请相应地修改命令。

这会创建一个名为/opt/lampp的目录，XAMPP安装于其中。要启动XAMPP，首先，将目录更改到新创建的目录。



```
# cd lampp
```

执行如下的命令，以启动XAMPP（启动Apache和MySQL）。

```
# ./lampp start
```

将会看到如下的信息。

```
Starting XAMPP for Linux 1.8.0...
XAMPP: Starting Apache with SSL (and PHP5)...
XAMPP: Starting MySQL...
XAMPP: Starting ProFTPD...
XAMPP for Linux started.
```

要测试Web服务器是否运行，打开一个Web浏览器并且输入<http://localhost/xampp/index.php>。XAMPP服务的菜单应该会显示出来，如图1-1所示。

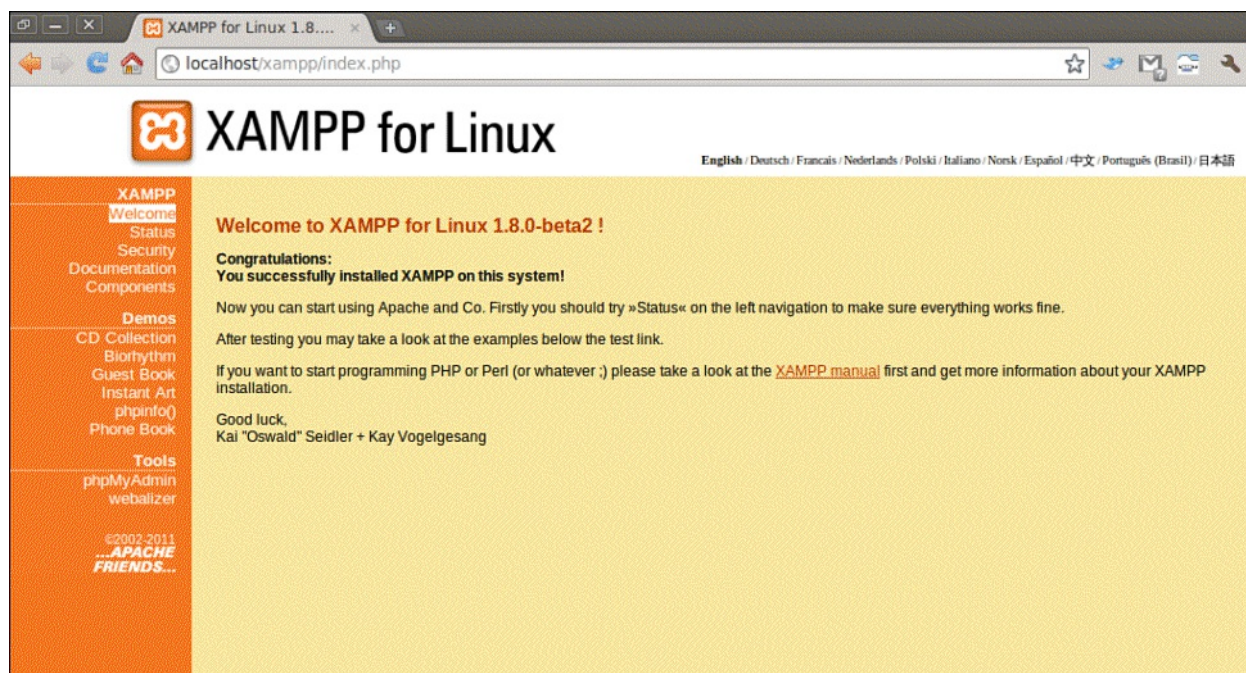


图1-1 XAMPP的菜单页面

所有要做的就是这些，XAMPP已经在你的机器上安装了Apache、

PHP和MySQL，并且，你可以看到服务的状态。在浏览<http://localhost/xampp/index.php>的时候，你可以通过左边列出的链接了解更多的相关信息。

要停止XAMPP及其服务，可以在任何时候通过命令行执行如下命令。

```
# /opt/lampp/lampp stop
```

确保阅读本章1.5节，了解有关保护安装了XAMPP的机器的更多信息（即使该机器只是用于开发）。

## 1.3 在Windows上安装XAMPP

随书光盘中的XAMPP安装文件已经经过了测试，并且适用于从Windows XP到Windows 7的多种Windows操作系统。Windows较早的版本则不支持。要使用随书光盘中的XAMPP安装文件，首先将光盘插入光驱，它应该会自动播放。如果没有，双击“我的电脑”下的光驱图标，并且找到包含XAMPP安装文件的目录。

找到随书光盘上的XAMPP文件，或者从<http://www.apachefriends.org/en/xampp-windows.html> 下载了最新版本之后，双击该文件以启动带向导的安装程序。

由于Windows操作系统发布版本的细微差别，并且由于Windows机器上可能安装了不同的安全策略和程序，如果任何安装步骤没有顺利进行，请访问位于<http://www.apachefriends.org/en/faq-xampp-windows.html> 的针对Windows用户的XAMPP FAQ。

首先会请你选择语言，English是默认的选择。在选择了语言并单击OK按钮之后，你将会看到安装程序的欢迎界面，如图1-2所示。



图1-2 XAMPP安装主界面

**注意：**

在编写本书的时候，XAMPP的版本号是1.7.7，因此，安装向导显示了该版本号。最新的版本会使用不同的版本号，但是，安装过程是类似的。

点击Next按钮以继续安装过程。就像大多数基于向导的安装一样，在进行下一步之前，安装程序要求你选择一个安装位置和一些安装选项。XAMPP的安装也没什么不同，你应该保留默认的安装位置，以及默认的安装选项，并且点击Next按钮继续进入下一个界面。此时，安装过程自动进行，如图1-3所示。

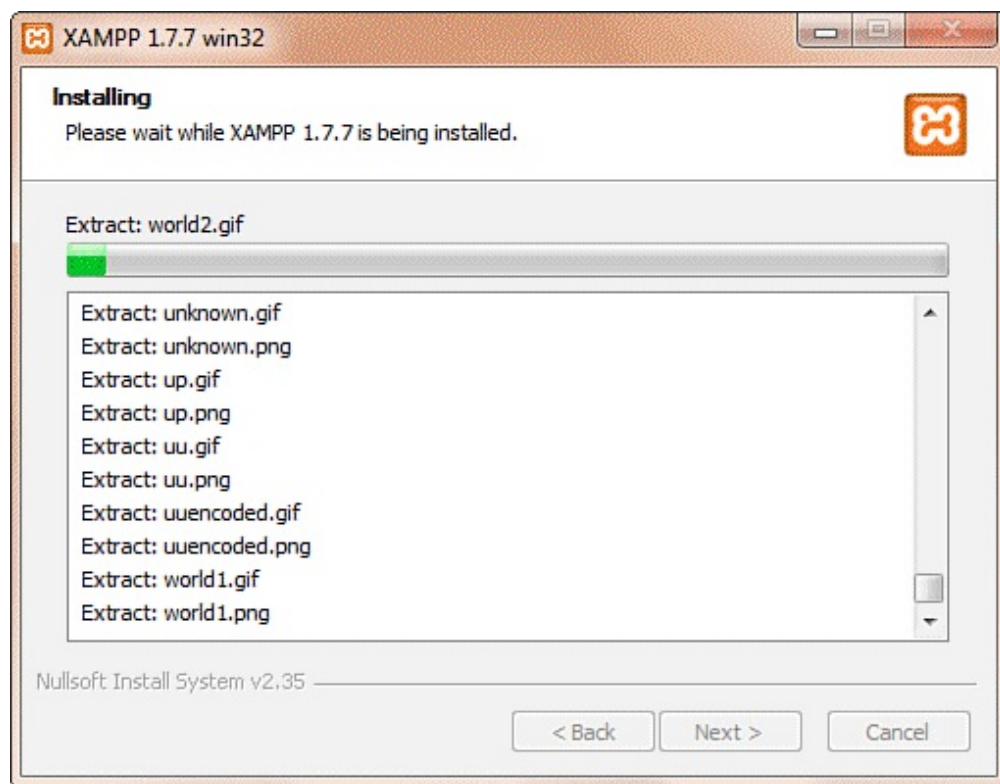


图1-3 XAMPP安装继续，解压文件

安装过程完成之后，安装程序会提示你当前状态；单击Finish按钮完成安装。在XAMPP安装过程结束之前，它询问你是否想要启动Control Panel，以管理所安装的服务，如图1-4所示。

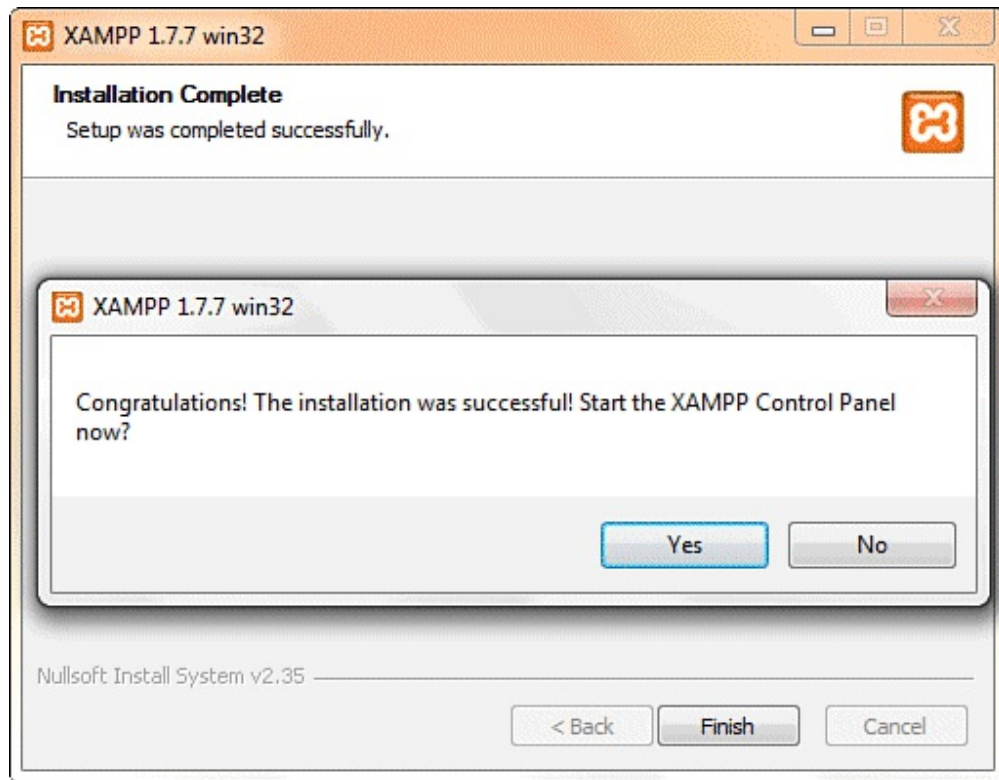


图1-4 XAMPP安装完成

XAMPP Control Panel使你能够通过一次鼠标单击来启动并停止在机器上运行的Apache和MySQL服务器进程，如图1-5所示。如果你只是为了进行开发而在本地机器上运行这些服务器进程，只有当你需要它们的时候，才想要打开它们；Control Panel允许你快速地做到这一点。



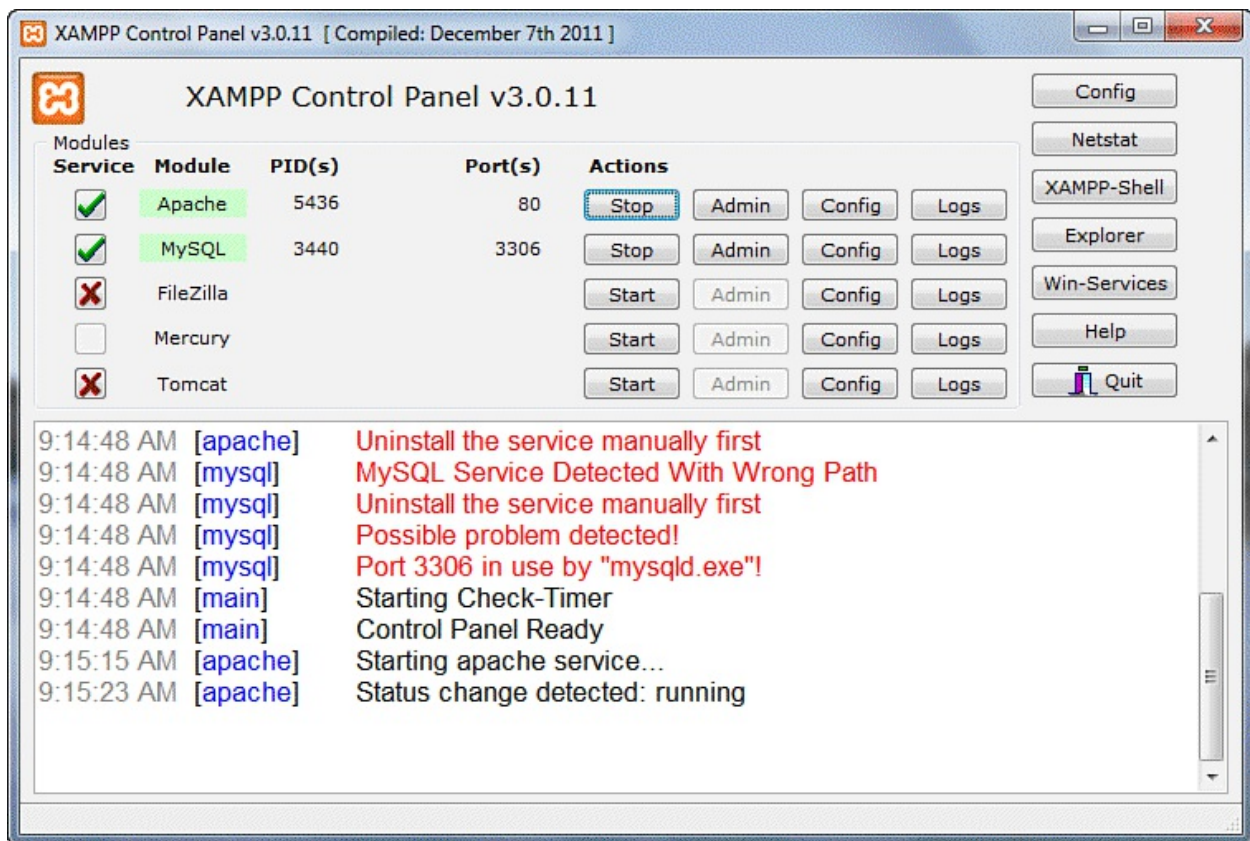


图1-5 XAMPP Control Pane

要测试Web服务器是否运行，打开Web浏览器并且输入 <http://localhost/xampp/xampp.php> 。应该会显示XAMPP菜单，如图1-6所示。



图1-6 XAMPP菜单页面

所有要做的就是这些，XAMPP已经在你的机器上安装了Apache、PHP和MySQL，并且，你可以看到服务的状态。在浏览 <http://localhost/xampp> 的时候，你可以通过左边列出的链接了解更多的相关信息。确保阅读本章1.5节，了解有关保护安装XAMPP的机器的更多信息（即使该机器只是用于开发）。



## 1.4 在Mac OS X上安装XAMPP

随书光盘中的XAMPP安装文件已经经过了测试，并且适用于Mac OS X 1.4及其之后的各个版本。Mac OS X的较早的版本则不支持。

要使用随书光盘中的XAMPP安装文件，首先将光盘插入Mac上的光驱，它应该会自动播放。如果没有，双击“我的电脑”下的光驱图标，并且找到包含XAMPP安装文件的目录。

找到随书光盘上的XAMPP文件，或者从<http://www.apachefriends.org/en/xampp-mac.html> 下载了最新版本之后，双击该文件以启动带向导的安装程序。你将会看到如图1-7所示的界面。



图1-7 在Mac上安装XAMPP的说明

在编写本书时，针对Mac用户的XAMPP的版本号是1.7.3，因此，这个安装程序包显示了该版本号。随后的版本将具有不同的文件名，因此，请相应地修改命令。

按照界面上的指示，将XAMPP文件夹拖曳到Applications文件夹中。在复制了文件夹和文件之后，可以在/Applications/XAMPP文件夹中找到XAMPP Control Panel的链接，如图1-8所示。

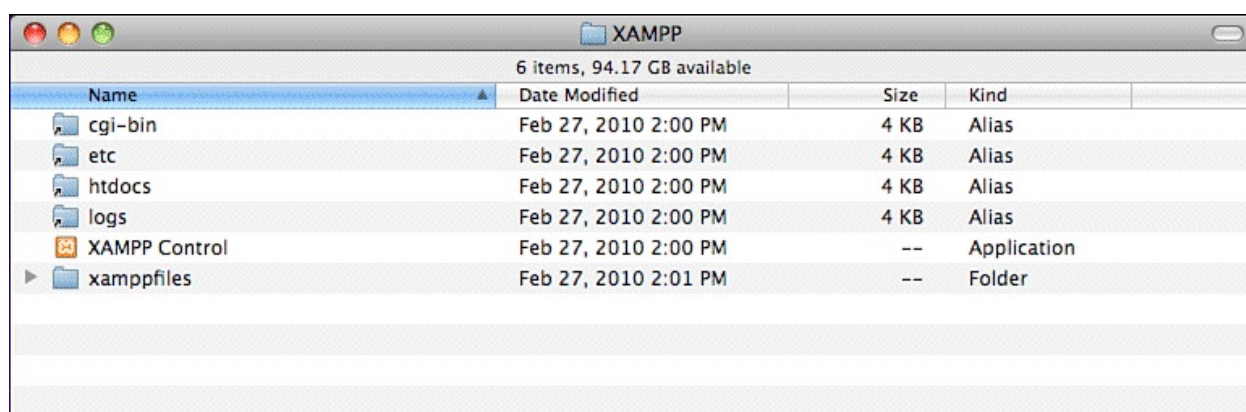


图1-8 找到XAMPP Control Panel的链接

双击该链接以打开XAMPP Control Panel，通过该面板我们可以在机器上启动或停止Apache和MySQL服务器进程。如果你只是为了进行开发而在本地机器上运行这些服务器进程，只有当你需要它们的时候，才想要打开它们；Control Panel允许你快速地做到这一点。

要测试Web服务器是否运行，打开Web浏览器并且输入<http://localhost/xampp/index.php>。XAMPP服务的菜单应该会显示出来，如图1-9所示。

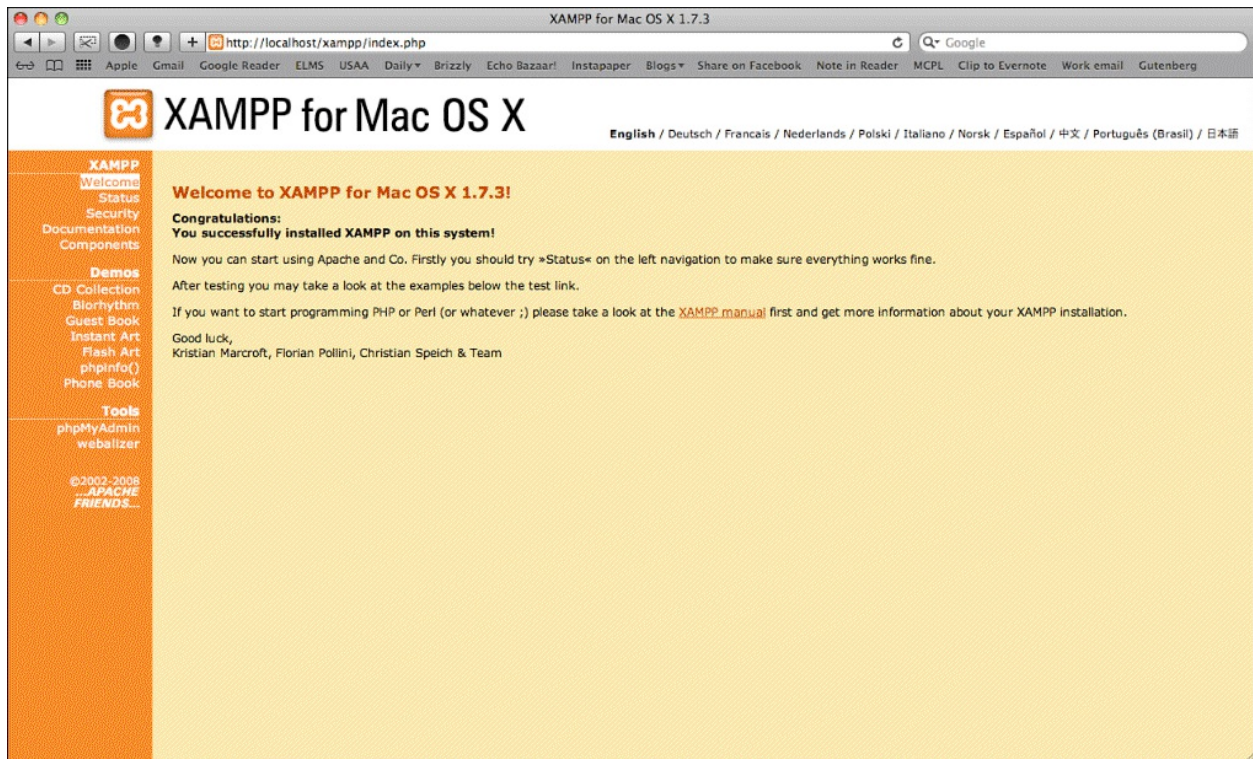


图1-9 XAMPP菜单页面

所有要做的就是这些，XAMPP已经在你的机器上安装了Apache、PHP和MySQL，并且，你可以看到服务的状态。在浏览<http://localhost/xampp/index.php>的时候，你可以通过左边列出的链接了解更多的相关信息。确保阅读本章1.5节，了解有关保护安装XAMPP的机器的更多信息（即使该机器只是用于开发）。

## 1.5 让XAMPP更安全

XAMPP的主要目的是提供一种快速容易的方式，以便在开发环境中安装Apache、MySQL和PHP。这种快速而容易的安装的代价之一，就是安全设置不完整，或者说，至少将确定安全设置是否足够重要的权利交给了用户。

使用最新的安装，会有如下的一些潜在的安全问题。

- MySQL管理员用户没有密码（你可以使用一个空白的密码）。
- 以同样的用户运行MySQL和Apache（nobody，这只针对Linux/UNIX和Mac操作系统）。
- 一些服务是可以通过该网络访问的，除非你通过自己的防火墙明确地禁止访问。

然而，XAMPP为每个操作系统都提供了一种工具，通过它，我们可以在一种开发环境中运行和完成增强XAMPP系统安全性的过程。

- 在Linux/UNIX上，通过输入如下命令来运行该工具。

```
# /opt/lampp/lampp security
```

- 在Windows上，通过在Web浏览器中访问<http://localhost/xampp/index.php>，并从左侧的导航菜单中选择Security，从而打开Security Console。
- 在Mac上，通过输入如下命令来打开一个终端窗口。

```
# /Applications/XAMPP/xamppfiles/xampp security
```

## 1.6 故障排除

本章的所有步骤已经用本书附带的光盘里包含的软件版本测试过。如果遇到安装问题，首先检查是否严格按照本章给出的步骤进行。

然后，查看位于<http://www.apachefriends.org/>的XAMPP Web站点，以了解适用于这一安装包的FAQ。

如果这些过程仍然不管用，并且你想要尝试其他的第三方整体安装包，那么，请自行尝试WAMP和MAMP（在本章开始处提到）。你也可以尝试使用“较复杂”的安装方式，这需要使用后续3章中介绍的扩展信息。这些后续章节也提供疑难解答提示，以及可以帮助解决安装问题的其他站点的链接。

## 第2章 安装和配置MySQL

在本章中，你将学到：

- 如何安装MySQL。
- 运行MySQL的基本安全规则。
- 如何通过用户授权系统来使用MySQL。

这是与复杂安装相关的三章中的第一章，在此你将学习如何安装开发环境。如果你打算使用MySQL和PHP，我们首先要介绍MySQL的安装，因为在某些系统中编译PHP模块需要完成MySQL的安装才行。



## 2.1 MySQL的当前版本和未来版本

本章的安装说明针对的是MySQL Community Server 5.5.21，这是MySQL软件的当前产品版本。这个版本号可以读作“MySQL服务器软件的主版本5，次版本（小发布）5的第21次修订”。修订版和小的发布并不遵从既定的一系列发布计划。当对代码进行扩展和修复并且进行了彻底的测试后，MySQL AB就会用一个新的修订号或次版本号来发布一个新的版本。

当你购买本书的时候，可能次版本号已经变成了5.5.22或者更高。如果是这种情况，你可以阅读位于<http://dev.mysql.com/doc/refman/5.5/en/news-5-5-x.html> 的列表来了解安装或配置过程中的任何变化，那里的内容构成了本章的大部分内容。

尽管在两个次版本更新之间不可能所有的安装过程都要变化，但你还是应该养成习惯查看自己所安装和维护的软件的更新日志。如果你阅读本书的时候，确实出现了一个次版本的变化，但更新日志中并没有提到安装的变化，你只需要用心记下，并且当出现在本书的安装说明和相应的图时，用新的版本号替代就行了。

## 2.2 如何获取MySQL

MySQL AB是负责开发、维护和发布MySQL数据库的公司的名字，经过一系列的收购之后（Sun Microsystems收购了MySQL AB，Oracle公司收购了Sun Microsystems），现在，数据库巨人Oracle拥有MySQL。然而，该软件的MySQL Community Edition版本一直保持开源，它是由开源开发者支持的，并且可以在MySQL的Web站点<http://www.mysql.com/> 免费获取。所有平台的二进制发布，用于Mac OS X的安装程序包，以及用于Linux/UNIX平台的RPM和源代码文件，都可以获得。

### 提示：

Linux和Mac OS X发布通常会包含MySQL软件的某个版本或其他开源的MySQL软件，尽管在当前的发布版本之后通常会有几个修订版或者小版本。

本章的安装说明基于正式发布的MySQL 5.5.x Community Server版本。所有的文件都可以从<http://dev.mysql.com/downloads/mysql/5.5.html>下载到，并且在编写本书的时候所采用的当前版本也可以在下载资料中找到。



## 2.3 在Linux/UNIX上安装MySQL

不管你是使用RPM还是二进制代码安装，在Linux/UNIX上安装MySQL都比较简单。如果你通过RPM安装，MySQL AB提供了专门平台的RPM，例如针对运行在不同类型的处理器（如32位或64位的x86）上的SuSE Linux或一般的Linux。

对于RPM的一个最小的安装，你需要如下来自下载页面<http://dev.mysql.com/downloads/mysql/5.5.html> 的两个文件（或者从本书随书光盘获取的文件）。

- **MySQL-server-type - VERSIONNUMBER.PLATFORM.rpm** —The MySQL server
- **MySQL-client-type - VERSIONNUMBER.PLATFORM.rpm** —The standard MySQL client libraries

要执行最小的RPM安装，只需要输入如下命令。

```
# rpm -i MySQL-server-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

### 提示：

把文件名中的VERSIONNUMBER替换为你所下载的实际版本，并且把PLATFORM替换为你所使用的平台的缩写。例如，当前的针对通用的Linux发布的MySQL 5.0 Server RPM，叫做MySQL-server-standard-5.0.51a-0.sles10.i586.rpm，而客户端库RPM叫做MySQL-client-standard-5.0.51a-0.sles10.i586.rpm。

对于Debian包的安装，你需要位于

<http://dev.mysql.com/downloads/mysql/5.5.html> 的页面中的\*.deb文件（或者从本书随书光盘中获取）。然后，在命令提示行中输入如下命令。

```
# dpkg -i mysql-VERSION-debian6.0-i686.deb
```

另一种轻松并且很常见的安装方法是通过一个二进制代码发布包来安装。这个方法需要gunzip和tar工具解压缩并拆包发布，并且还需要能够在系统上创建组和用户。二进制代码发布安装过程中的第一组命令，就是添加一个组和一个用户并且拆包发布，如下所示。

```
# groupadd mysql
# useradd -r -g mysql mysql
# cd /usr/local
# tar zxvf /path/to/mysql-VERSION-PLATFORM.tar.gz
```

提示：

把文件名中的VERSION-PLATFORM替换为你实际所下载的版本。例如，当前的MySQL 5.5通用Linux二进制代码发布叫做mysql-5.5.21-1- linux2.6.i386.tar。

接着，指令使用一个较短的名字创建一个符号链接。

```
# ln -s mysql-VERSION-PLATFORM mysql
# cd mysql
```

一旦拆包，README文件和INSTALL文件将会根据你所选择的MySQL版本来完成剩下的安装过程。通常会用到如下的一组命令。

```
# scripts/mysql_install_db --user=mysql
# chown -R root .
# chown -R mysql mysql_data
# chgrp -R mysql .
# bin/mysqld_safe --user=mysql &
```

我们现在已经准备好启动MySQL服务器，所以跳转到2.7“基本安全

规则”来学习如何添加密码和用户。如果在安装中碰到任何问题，请查阅2.6“安装故障排除”一节。

## 2.4 在Mac OS X上安装MySQL

Mac OS X下的MySQL安装过程相当简单，MySQL AB的开发者已经为Mac OS X创建了一个安装包。到位于<http://dev.mysql.com/downloads/mysql/5.5.html>的MySQL下载页面并且找到Mac OS X（或者使用随书光盘中的文件）。如果你访问该站点，确保下载了适合你的系统的DMG：无论你使用的是Mac OS X 10.5或10.6版，或者32位或64位的系统。当你有了该文件，双击DMG文件夹。在打开了DMG存档之后，你会看到一个文件夹其中有一些文件，如图2-1所示。

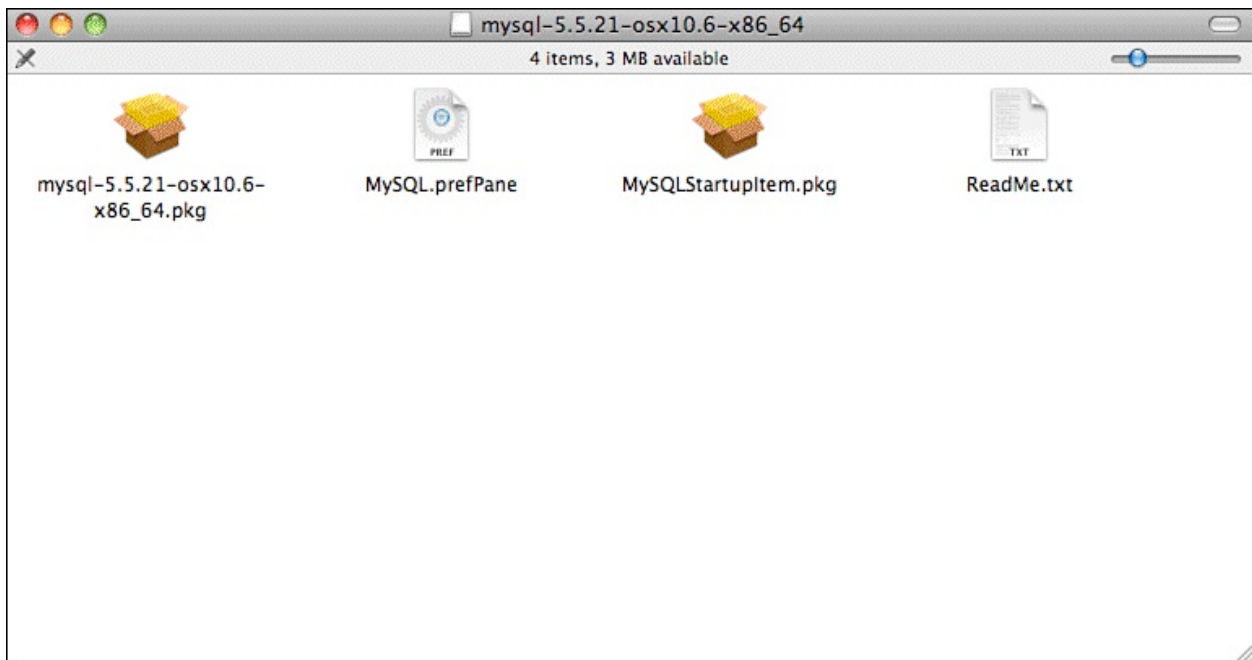


图2-1显示MySQL DMG文件夹的内容

当下载结束，拆包安装文件并且双击\*.pkg文件。后续的安装步骤完成如下过程。

1. MySQL安装包自动启动（如图2-2所示）。点击Continue跳转到下一步。

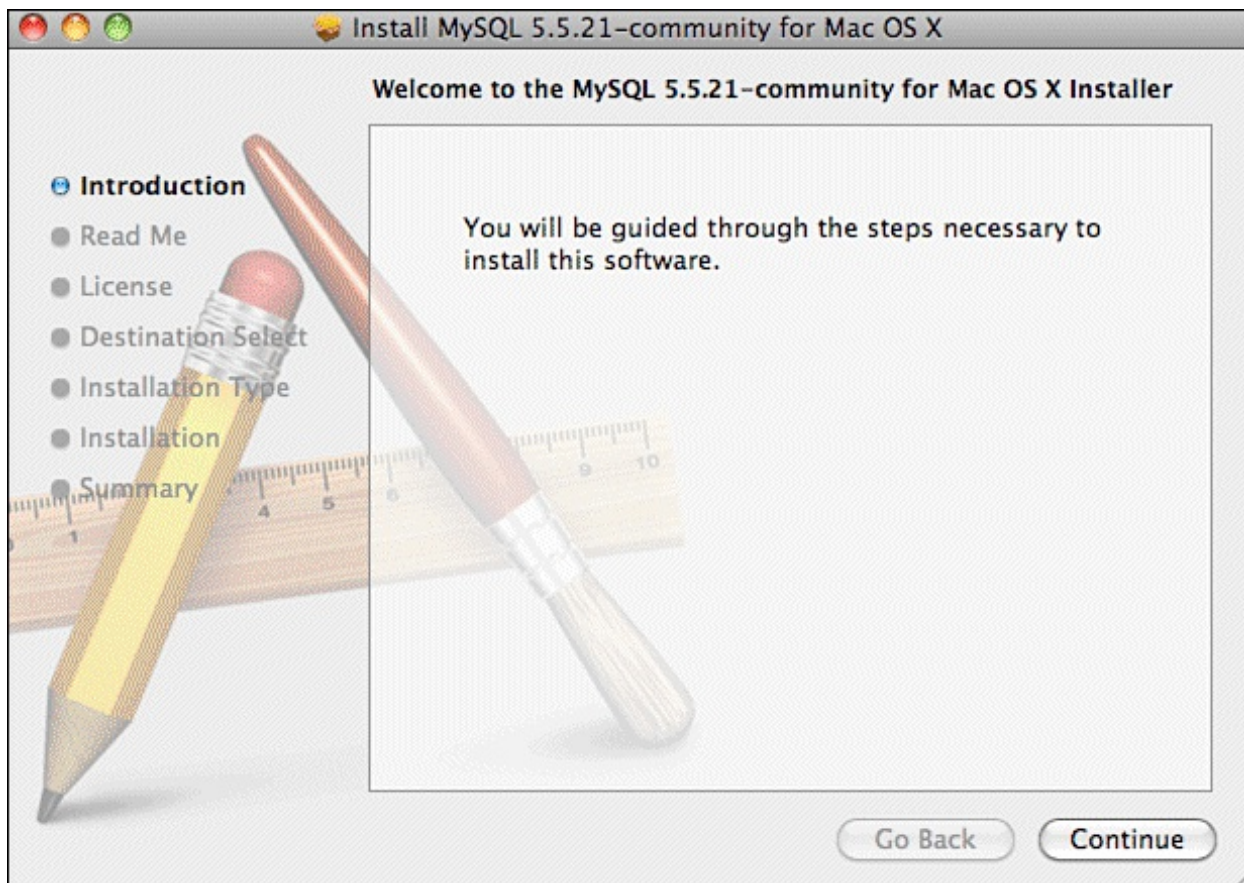


图2-2 Mac的MySQL安装程序已经启动

2. 接下来的几个界面将会包含与安装以及MySQL授权许可相关的通用信息。阅读这些界面的内容并单击Continue按钮通过它们。

3. 在通过了信息和授权许可界面之后，必须选择一个安装目标。从图2-3所示的界面中选择相应的驱动器，然后单击Continue按钮。

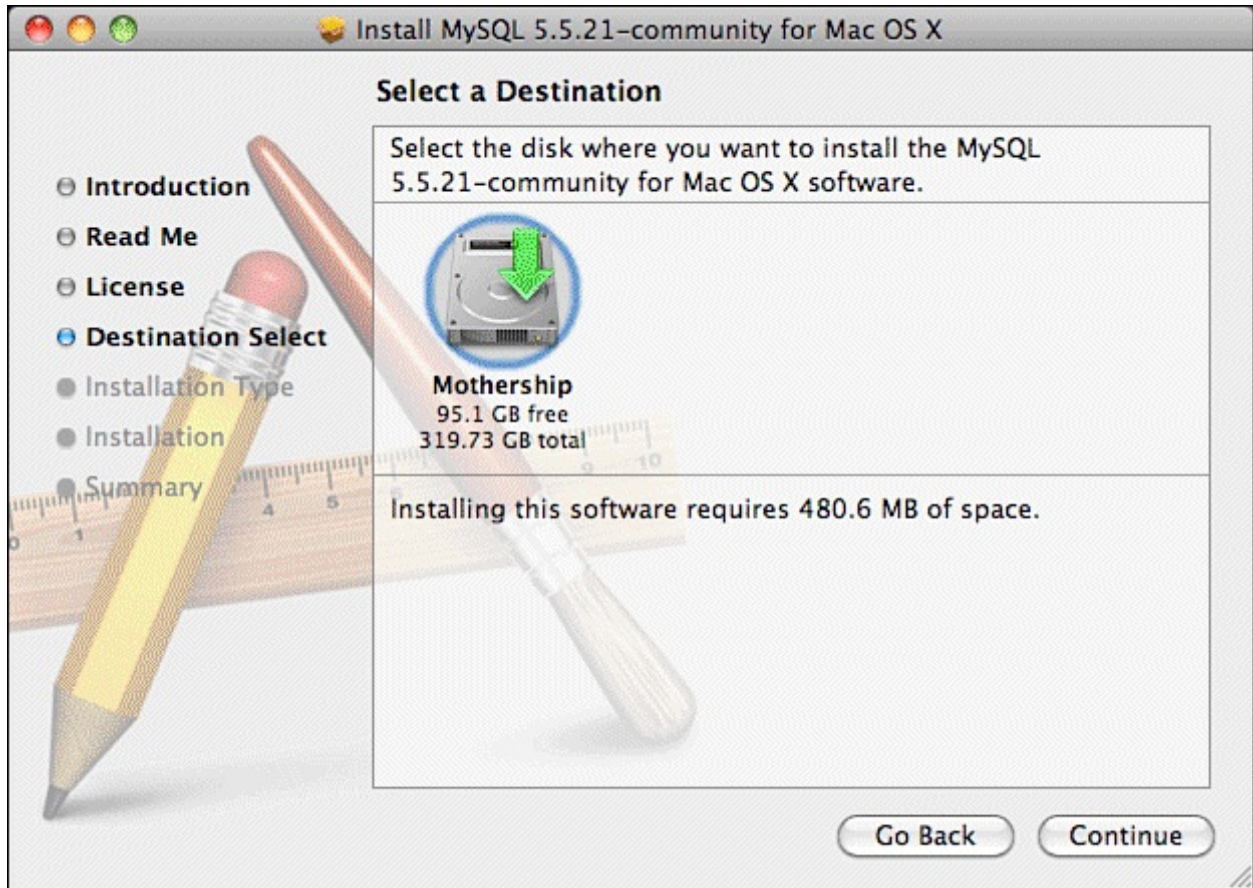


图2-3 选择一个安装位置

4. 下一步会验证安装位置选择，并且需要单击Install按钮来继续。此时，在这个安装过程继续进行之前，可能提示你输入管理员用户名和密码。一旦继续开始，就让此过程运行，直到安装完成，如图2-4所示。



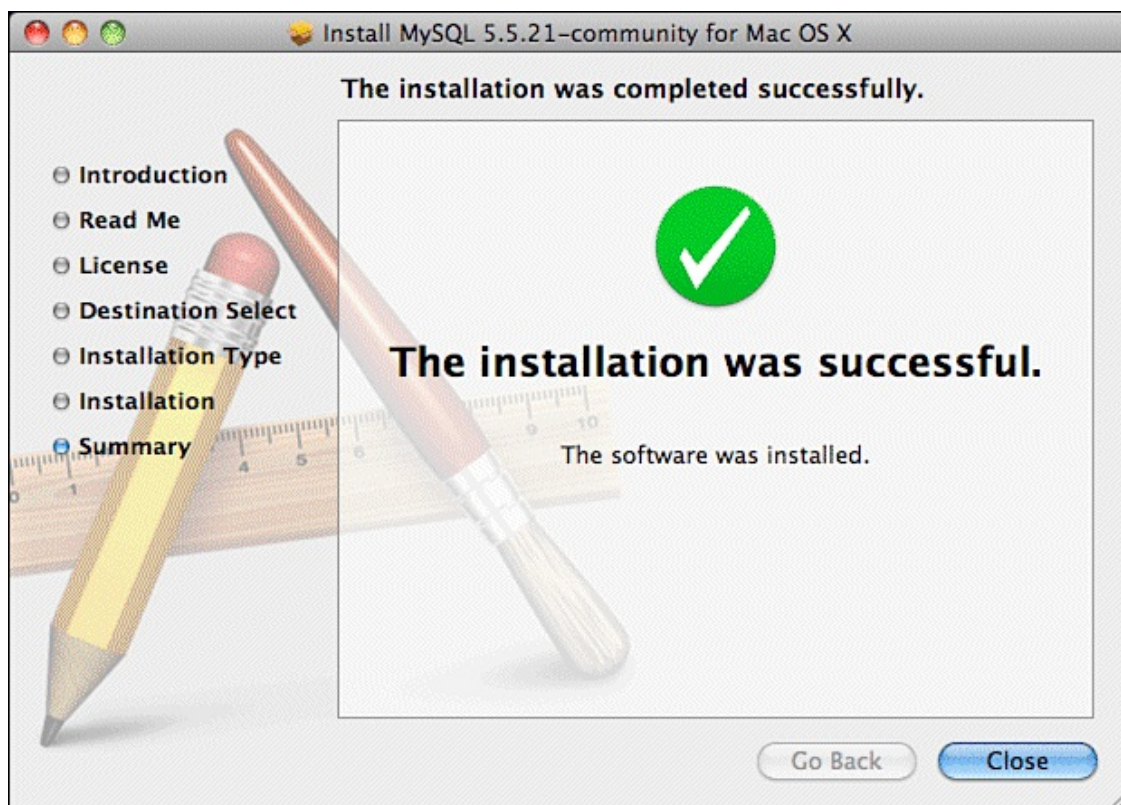


图2-4 MySQL已经安装

安装包中还包含了MySQL Startup Item安装程序。如果希望MySQL在系统启动的时候自动启动，就安装这个附加的包。MySQL Startup Item的安装遵从刚才所描述的标准安装方法，即双击\*.pkg文件，选择一个目标盘，然后让安装过程运行到完成。安装了MySQL Startup Item之后，在命令行窗口中使用如下的命令来启动MySQL。

```
# sudo /Library/StartupItems/MySQLCOM/MySQLCOM start
```

当你试图启动MySQL的时候，可能要求你输入管理员密码。在启动了MySQL之后，按下Ctrl+D退出命令行窗口。启动了MySQL之后，我们可以跳到2.7“基本安全规则”一节。如果在安装中碰到任何问题，请查阅2.6“安装故障排除”一节。

## 2.5 在Windows上安装MySQL

Windows上的MySQL安装过程使用一个标准的Microsoft Windows安装程序（Microsoft Windows Installer，MSI）文件来完成在Windows XP、Windows Server 2003、Windows Vista或Windows 7机器上MySQL的安装和配置过程。到位于

<http://dev.mysql.com/downloads/mysql/5.0.html> 的MySQL下载页面，并且找到标题为“Windows Downloads”的小节。下载Windows Essentials文件，其扩展名为\*.msi。下载完这个文件，双击文件开始安装过程。

如下的步骤给出了使用MySQL AB的Windows Essentials安装程序在Windows上安装MySQL 5.0.20的细节。不管你的Windows环境是什么，安装过程将遵从同样的步骤。

### 注意：

Windows用户也可以使用ZIP Archive版本。如果你想要安装ZIP Archive版本，确保阅读位于<http://dev.mysql.com/doc/refman/5.5/en/windows-choosing-package.html> 的MySQL手册的说明和介绍。

直接进入安装过程，按照如下的步骤进行。

1. 双击\*.msi文件开始安装过程。你将会看到安装向导的第一个界面，如图2-5所示。单击Next按钮继续。



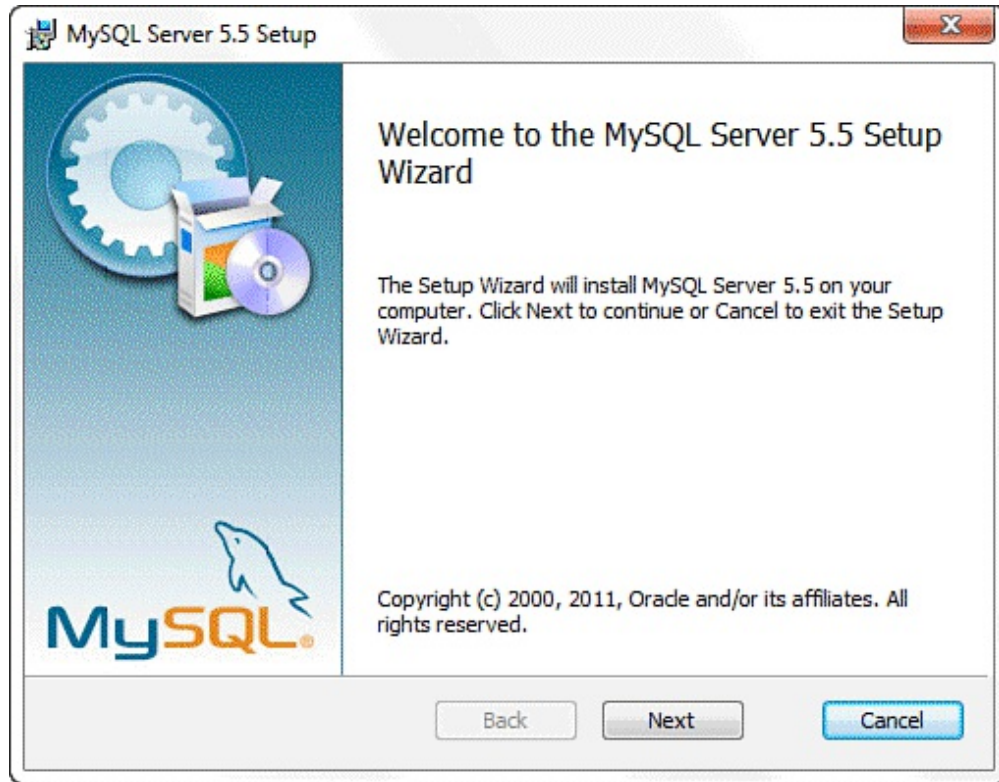


图2-5 Windows下MySQL安装向导的第一步

2. 选择安装方式。Typical（典型）、Complete（完全）或 Custom（自定义）（如图2-6所示）。Custom选项允许挑选和选择要安装的MySQL组件，而Complete选项则会安装MySQL的所有组件，包括文档和工具包套件。Typical安装方式则适合于大多数用户，因为它包括了用来对MySQL进行一般性管理的客户机、服务器和众多工具。选择Typical安装方式，并且单击Next按钮继续。

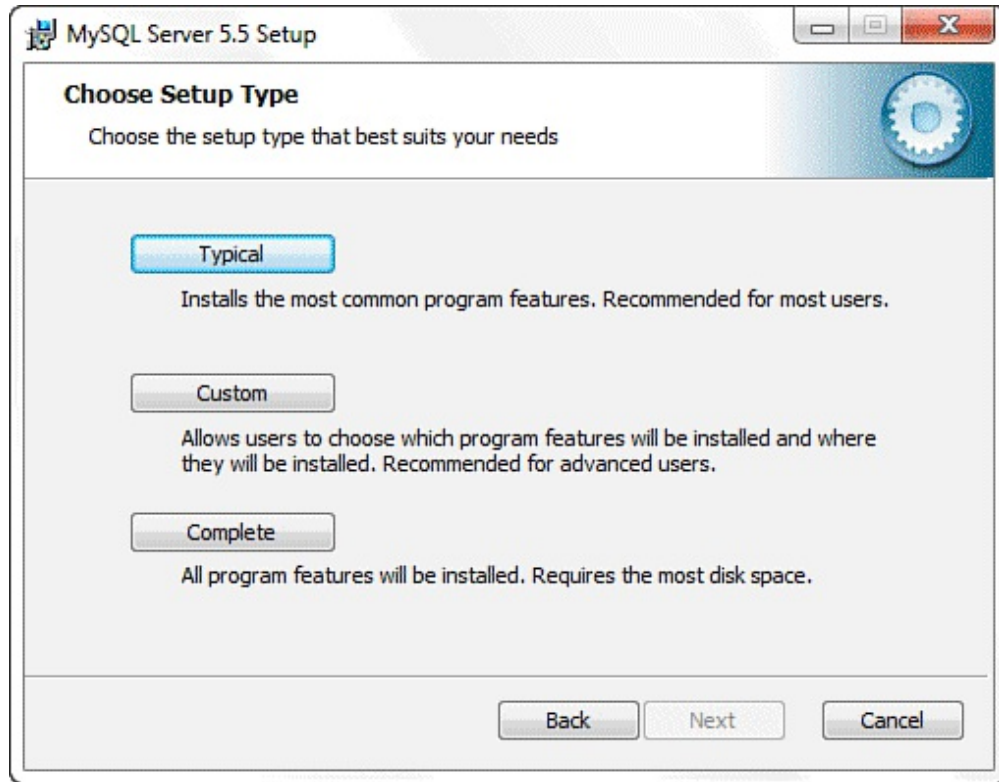


图2-6 选择一种安装方式

3. 确认在下一个界面中的选择并且单击**Install**按钮继续。安装过程将负责把文件安装到正确的位置。

4. 当安装过程完成后，可以选择继续MySQL Configuration Wizard（MySQL配置向导）。强烈推荐运行这个向导，因为它会创建一个自定义的my.ini文件，它根据你的具体需求而设置。要继续进行MySQL配置向导，选中Configure the MySQL Server Now复选框，并且单击**Finish**按钮，如图2-7所示。

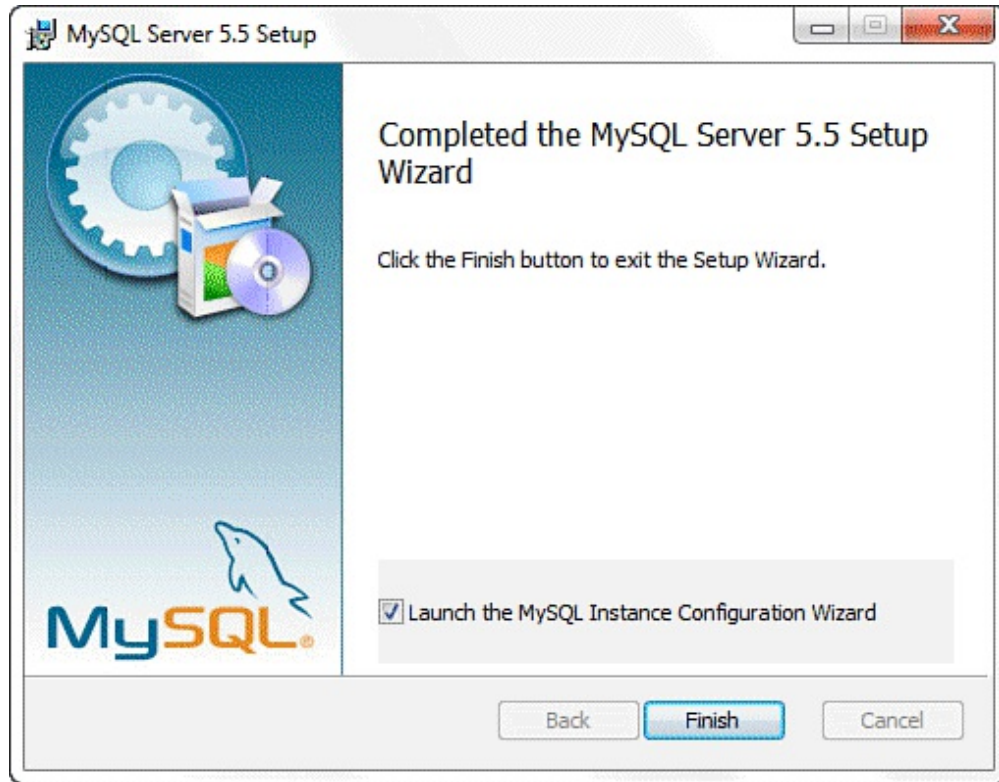


图2-7 现在，MySQL已经安装了，继续执行MySQL Configuration Wizard

5. 你将看到配置向导的欢迎界面，单击Next按钮来继续向导的下一个步骤。你将会看到服务器配置的两个选项：Detailed和Standard。我们使用Detailed Configuration选项，这样可看到可用的所有选项。如果决定选择Standard Configuration选项，必须手动修改文件my.ini以达到想要的配置。选择Detailed Configuration单选按钮，然后单击Next按钮继续。

6. 必须在如图2-8所示的界面中做出下一个选择。在这个步骤中，你要选择所运行的机器类型：Developer Machine、Server Machine或者Dedicated MySQL Server Machine。在这个界面上的选择决定了所用的内存、硬盘和处理器的分配。如果你为了测试而在个人电脑上使用MySQL，则选择Developer Machine选项。如果MySQL所运行的机器上

还有其他的服务器软件，并且比你在个人电脑上运行MySQL要占用更多的系统资源，那就选择Server Machine选项。如果MySQL是机器上所运行的主要服务，并且可以使用大量的系统资源，选择Dedicated MySQL Server Machine选项。在做出选择之后，单击Next按钮继续。

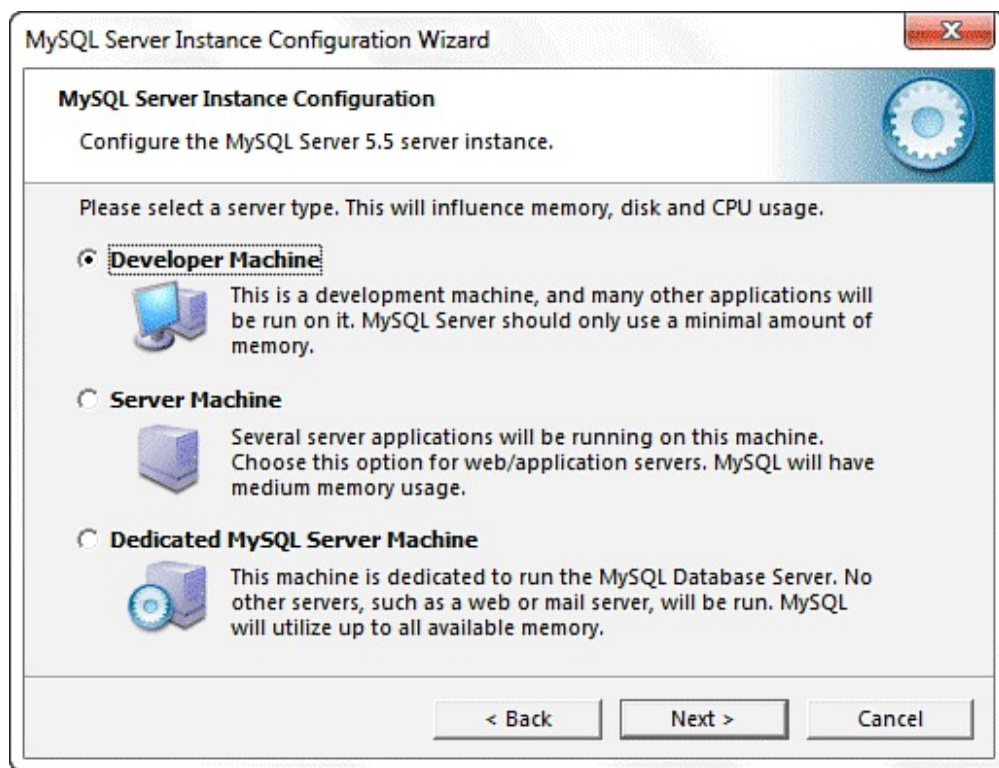


图2-8 选择服务器类型作为MySQL配置的一部分

7. 下一个配置选项适合于数据库应用。这些选项是Multifunctional Database、Transactional Database和Non-Transactional Database Only。对于Multifunctional Database，InnoDB和MyISAM存储引擎二者之间可以平均地分配资源。Transactional Database也支持InnoDB和MyISAM，但是大多数服务器资源倾向于InnoDB。Non-Transactional Database Only则不支持InnoDB，并且把所有的资源应用于MyISAM。除非你确切地知道自己的数据库使用哪种存储引擎，否则选择Multifunctional Database单选



按钮并单击Next按钮继续。

8. 如果已经选择了一个数据库使用选项，其中包含了InnoDB存储引擎，配置过程的下一步就允许配置硬盘位置和存储阈值。默认的情况如图2-9所示，可以单击Next按钮继续简单地确认默认配置，或者可以修改这些设置，然后单击Next按钮继续，从而让自定义设置起作用。

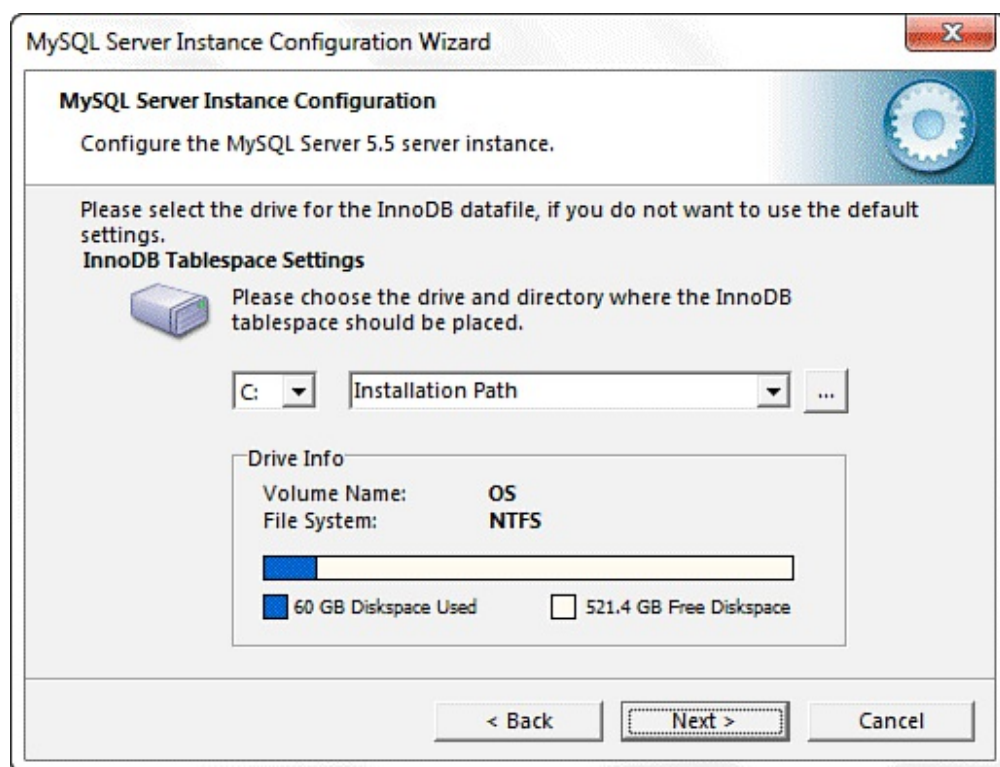


图2-9 为InnoDB存储引擎调整硬盘使用选项

9. 下一个配置选项决定了MySQL服务器的并发连接数。你的设置取决于Web站点或应用程序所使用的数据流和数据库的数量。默认的设置是Decision Support (DSS)/OLAP，最多100个并发连接，假定平均值为20个。Online Transaction Processing (OLTP)选项的最多并发连接数是500个，而Manual设置允许我们从下拉列表中选择一个数值或者自己键入一个数值。做出自己的选择并单击Next按钮继续。

10. 配置过程中的下一步是Networking Options界面。在这里，我们可以激活或者关闭TCP/IP网络，并且可以配置连接到MySQL的端口号，默认是3306，实际上我们可以使用任何没有使用的端口。这个界面中的另一个选项可以打开或关闭严格模式，推荐选择打开严格模式，除非你知道要改变什么。参见<http://dev.mysql.com/doc/refman/5.5/en/server-sql-mode.html> 了解更多信息。做出自己的选择并单击Next按钮继续。

**提示：**

别忘了修改防火墙规则以允许数据流从3306端口（或者任何你确定用于MySQL的端口）通过。

11. Networking Options界面之后是Character Set选项。默认的选项是Standard Character Set，这使得整个数据库都采用Latin1。也可以选择“Best Support for Multilingualism”选项，这使得UTF8作为字符集。UTF8允许我们在一个单个字符集中存储多种语言。如果你想要使用某个特定的字符集，选择“Manual Selected Default Character Set”单选按钮，然后从下拉列表中选择相应的字符集。在做出选择之后，单击Next按钮继续。

12. 推荐把MySQL作为一项服务安装。选中“Install as Windows Service”复选框并且为服务选择一个名字。“Launch the MySQL Server Automatically”复选框是可选的。还可以选择把MySQL bin目录添加到Windows PATH，以便更容易地从命令提示符窗口调用MySQL，如果这种情况很适合你，就选中这个复选框。完成了选择之后，单击Next按钮继续。

13. Security Options配置界面也是所有配置界面中最重要的一個。如图2-10所示，使用这一配置界面来设置一个root用户的密码。输入密码两次以便确认。不要选中“Enable Root Access From Remote Machines”复选框，除非你真的知道在做什么。通常，root用户连接只允许来自服务器本地。另外，可以创建一个匿名用户，但出于安全原因，并不建议这么做。完成了这个界面中的配置选项之后，单击Next按钮继续。

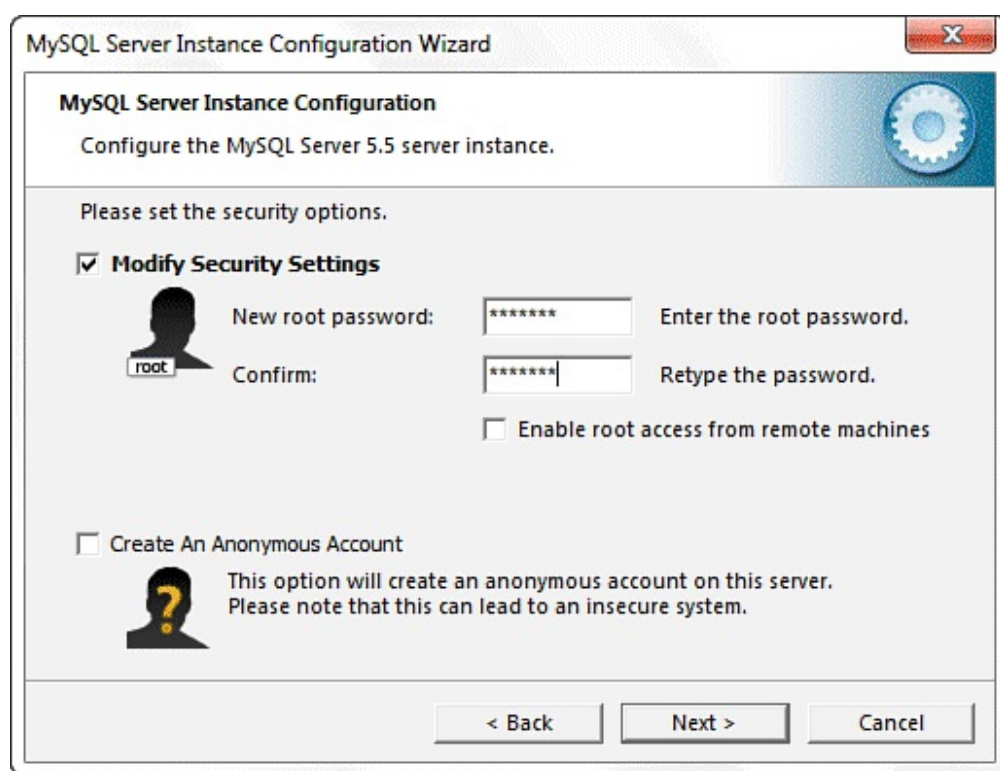


图2-10 通过MySQL配置，为root创建一个密码

14. 配置过程还保留了另一个更多的步骤，单击Execute按钮可以开始这一步骤。在向导完成了各种配置步骤之后，将会看到如图2-11所示的一个配置界面，表示配置文件已经创建，并且MySQL服务已经启动。单击Finish按钮结束向导。

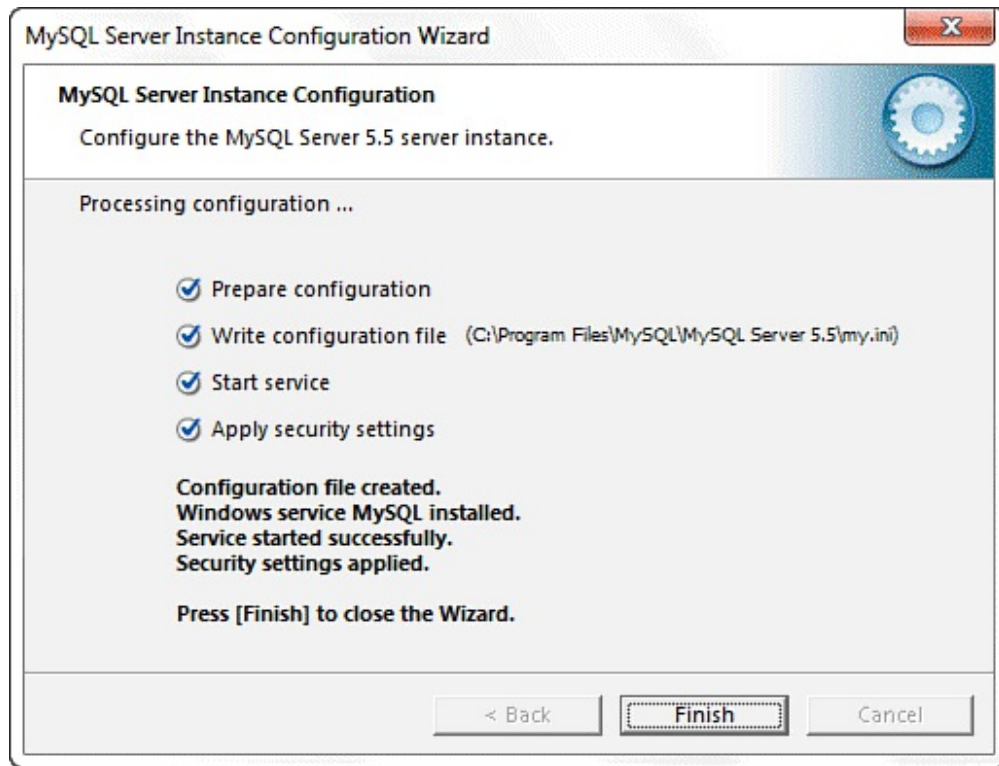


图2-11 MySQL配置向导完成其任务

安装和配置向导的完成会产生一个运行的MySQL服务，并且在C:\Program Files\MySQL\MySQL Server 5.5\目录下产生一个自定义的my.ini文件。

**提示：**

可以使用任何文本编辑器来手动编辑my.ini文件，在修改之后必须重新启动MySQL服务器。

现在MySQL已经启动了，跳转到2.7节。如果在安装中碰到任何问题，请查阅2.6节。



## 2.6 安装故障排除

如果在MySQL安装过程中碰到任何问题，首先应该查看MySQL手册的附录A，它位于<http://dev.mysql.com/doc/refman/5.5/en/problems.html>。

下面只是一些常见的安装问题。

- 在Linux/UNIX和Mac OS X上，不正确的权限许可不允许你启动MySQL守护进程。如果情况是这样，确保你已经把所有者和组修改为与安装说明中相匹配的那些内容。
- 如果在连接到MySQL的时候看到了消息Access denied，请确保你使用了正确的用户名和密码。
- 如果看到了消息Can't connect to server，请确保MySQL守护进程在运行。

如果在阅读了MySQL手册的附录A之后仍然有问题，那么发送邮件到MySQL邮件列表（参见<http://lists.mysql.com/> 了解更多信息）可能会知道结果。你也可以从MySQL AB购买支持协议。

## 2.7 基本安全规则

不管是在Windows、Linux/UNIX还是Mac OS X上运行MySQL，也不管是管理自己的服务器还是使用Internet服务提供商所提供的系统，你都必须理解基本安全规则。如果你通过Internet服务提供商访问MySQL，需要注意几个服务器安全性的方面。例如，一个非root用户不能够修改或绕过身份验证。不幸的是，很多Internet服务提供商对安全规则毫不在意，让他们的客户暴露在外，并且在很大程度上，他们没有意识到风险。

### 2.7.1 启动MySQL

增强MySQL的安全性从服务器的启动阶段就开始了。如果你不是服务器的管理员，就不能改变服务器的安全设置，但是，你肯定可以查看服务器的安全性，并且向Internet服务提供商报告弱点。

如果MySQL安装在Linux/UNIX或Mac OS X上，主要关心的问题应该是MySQL守护程序的所有者，它不应该是root。把守护程序作为一个非root用户的进程运行，例如mysql或database，将会限制恶意用户获得访问服务器或者覆盖文件的能力。

#### 提示：

可以在Linux/UNIX或Mac OS X系统上使用ps（进程状态）命令来验证进程的所有者。

如果看到MySQL在系统上作为root用户运行，应立即联系你的

Internet服务提供商并且提出意见。如果你是一个服务器管理员，应该作为非root用户启动MySQL进程，或者在启动命令行里指定首选的用户名。

```
# mysqld --user=non_root_user_name
```

例如，如果你想要作为用户mysql运行MySQL，使用如下命令。

```
# mysqld --user=mysql
```

然而，启动MySQL的推荐方法是通过MySQL安装的bin目录中的mysqld\_safe启动脚本来进行。

```
# bin/mysqld_safe --user=mysql &
```

## 2.7.2 增强MySQL连接的安全

你可以以几种不同的方式连接到MySQL监视器或者其他MySQL应用程序，每种方式都有安全性风险。如果你的MySQL安装在自己的工作站上，和那些必须使用一个网络链接连接到他们的服务器的用户相比，你可以少些担忧。

如果MySQL安装在工作站上，最大的安全风险就是MySQL监视器或MySQL GUI管理工具没有关注到工作站的启动和运行。在这种情况下，任何人都可以利用并删除数据，插入假的数据，或者关闭服务器。如果你必须让工作站在一个公共领域内保持无监控状态，就使用一个带有密码的屏幕保护或锁定屏幕的机制。

如果MySQL安装在你的网络之外的一个服务器上，连接的安全性应该受到关注。就像任何通过Internet的数据传输一样，数据可能被截

获。如果传输是未加密的，截获数据的人就可以将它们拼接起来并使用信息。假设未加密传输的数据是你的MySQL登录信息，一个恶意者现在就可以伪装成你来访问数据库了。

防止这种情况发生的一种方法，就是通过一个安全的链接（例如，Secure Shell，即SSH）来连接到MySQL。通过SSH，所有到远程机器的传输和来自远程机器的传输都是加密的。类似地，如果你使用一个基于Web的管理界面，例如phpMyAdmin（参见<http://www.phpmyadmin.net/>以了解更多信息，注意，phpMyAdmin在第1章所介绍的基于XAMPP的快速安装中，已经安装过了），或者你的Internet服务提供商所使用的另一种工具，请通过一个安全的HTTP连接来访问该工具。

在下一节中，你将会了解到MySQL的权限系统，这有助于使服务器获得更深层次的安全保护。

## 2.8 MySQL权限系统简介

MySQL权限系统总是起作用的。当你第一次尝试连接MySQL服务器的时候，并且对于每一个后续的动作，MySQL都会检查以下3件事情。

- 你从哪里访问（你的主机）？
- 你说你是谁（你的用户名和密码）？
- 允许你做什么（你的命令权限）？

所有这些信息都存储在一个名为mysql的数据库中，当安装MySQL的时候，自动创建该数据库。在mysql数据库中，有如下几个和权限相关的表。

- **columns\_priv** —— 为一个表中的具体字段定义用户权限。
- **db** —— 为服务器上的所有数据库定义许可。
- **host** —— 定义连接到一个具体数据库的、可接受的主机。
- **procs\_priv** —— 为存储例程定义用户权限。
- **tables\_priv** —— 为一个数据库中的具体的表定义用户权限。
- **user** —— 为一个具体用户定义命令权限。

在本章中，当你向MySQL添加一些示例用户的时候，这些表将变得更为重要。现在，只需要记住这些表的存在，并且为了让用户完成操作，这些表中必须拥有相关的数据。

### 2.8.1 两步身份验证过程

正如你所了解的，在身份验证过程中，MySQL检查3件事情。和这3件事情相关的动作分如下两步执行。

1. MySQL查看你的连接所来自的主机，以及所使用的用户名和密码。如果主机允许连接，你的用户名对应的密码正确，并且用户名和分配给该主机的一个用户名匹配，MySQL就转到第二步。

2. 对于你尝试执行的任何一条SQL命令，MySQL验证你能够对该数据库、表和字段执行此操作。如果步骤1失败，你将会看到一个相关的错误，并且不能继续步骤2。例如，假设你使用一个用户名joe和一个密码abc123连接到MySQL，并且想要访问一个名为myDB的数据库。如果由于如下原因导致这些连接变量的任何一个不正确，你都会接收到一条类似如下的错误消息。

- 密码不正确。
- 用户名joe不存在。
- 用户joe不能从localhost连接。
- 用户joe能够从localhost连接，但不能使用myDB数据库。

你可能看到如下的一条错误消息。

```
# mysql -h localhost -u joe -pabc123 test
Error 1045: Access denied for user: 'joe@localhost' (Using password: YES)
```

如果带有密码abc123的用户joe允许从localhost连接到myDB数据库，MySQL将会在这个过程的第二个步骤中检查joe所能执行的操作。为了便于说明，假设jow允许查询数据但是不允许插入数据。事件和错误的序列就会如下所示。

```
# mysql -h localhost -u joe -pabc123 test
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12 to server version: 5.5.21-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SELECT * FROM test_table;
+----+-----+
| id | test_field |
+----+-----+
|  1 | blah      |
|  2 | blah blah  |
+----+-----+
2 rows in set (0.0 sec)

mysql> INSERT INTO test_table VALUES ('', 'my text');
Error 1044: Access denied for user: 'joe@localhost' (Using password: YES)
```

基于操作的许可在具有多层级管理的应用程序中很常见。例如，如果已经创建了包含个人财务数据的应用程序，你必须确保对记账级别的成员只赋予SELECT权限，而对具有安全许可的主管级成员赋予INSERT和DELETE权限。

在大多数情况下，当你通过一个Internet服务提供商访问MySQL的时候，只有一个用户和一个数据库可供使用。默认情况下，一个用户将能够访问该数据库中所有的表，并且允许执行所有的命令。在这种情况下，作为开发者，你的职责就是通过自己的编程开发出一个安全的应用程序。

然而，如果你是自己的服务器的管理员，或者Internet服务提供商允许你任意添加多个数据库和用户，并且可以修改自己的用户的访问权限，下面几个小节将带你学习如何做到这些。

## 2.8.2 添加用户

通过一个第三方应用程序来管理服务器，这为你提供了一个简单的方法来添加用户，只要使用一个类似向导的过程或一个图形化界面。然而，通过MySQL监视器添加用户并不难，尤其是如果你理解了MySQL所使用的安全检查点，这我们刚才已经学习过。

添加新用户的最简单的方法就是使用GRANT命令。作为root用户连接到MySQL，我们就可以使用一条命令来建立一个新用户。另一种方法是使用INSERT语句修改mysql数据库中所有相关的表，这需要我们知道用来存储许可的表的所有字段。第二种方法的效果和GRANT命令相同，但是比GRANT命令复杂得多。GRANT命令的简单语法如下。

```
GRANT privileges
ON databasename.tablename
TO username@host
IDENTIFIED BY "password";
```

下面是我们可以授予的一些常见的权限。如果需要完整的权限列表，请参考位于<http://dev.mysql.com/doc/refman/5.5/en/grant.html> 的MySQL手册的GRANT条目。

- **ALL** ——授予用户所有常见权限。
- **ALTER** ——用户可以改变（修改）表、列和索引。
- **CREATE** ——用户可以创建数据库和表。
- **DELETE** ——用户可以从表中删除记录。
- **DROP** ——用户可以删除表和数据库。
- **FILE** ——用户可以读取和写入文件，这个权限用来导入或转储数据。
- **INDEX** ——用户可以添加或删除索引。
- **INSERT** ——用户可以向表中添加记录。



- **PROCESS** ——用户可以查看并停止系统进程，只有可信任的用户才能拥有此权限。
- **RELOAD** ——用户可以使用FLUSH语句，只有可信任的用户才能拥有此权限。
- **SELECT** ——用户可以从表中选取记录。
- **SHUTDOWN** ——用户可以关闭MySQL服务器，只有可信任的用户才能拥有此权限。
- **UPDATE** ——用户可以更新（修改）表中的记录。

例如，如果你想要创建一个带有99hjc!5密码的用户john，他在名为myDB的数据库中的所有表上都有SELECT和INSERT权限，并且希望这个用户能够从任何主机连接，那么，使用如下命令。

```
GRANT SELECT, INSERT
ON myDB.*
TO john@"%"
IDENTIFIED BY "99hjc!5";
```

注意两个通配符\*和%的使用。这两个通配符用来代替值。在这个例子中，\*代替了数据库的全部表，而%代替了已知的世界中的所有主机的列表，这实际上是非常长的一个列表。

这里还有使用GRANT命令添加用户的另外一个例子。这次是添加一个带有密码45sdg11的用户jane，他在名为myCompany的数据库的一个名为employees的表上具有ALL权限。这个新的用户只能从一个特定的主机连接。

```
GRANT ALL
ON myCompany.employees
TO jane@janescomputer.company.com
IDENTIFIED BY "45sdg11";
```

如果你知道janescomputer.company.com有一个IP地址63.124.45.2，可以用这个地址来替代命令中的主机名部分，命令如下所示。

```
GRANT ALL
ON myCompany.employees
TO jane@janescomputer.company.com
IDENTIFIED BY "45sdg11";
```

添加用户的时候需要注意一点：总是使用密码并且确保这个密码是不易被破解的。

如果你使用GRANT命令来添加用户，改变会立即生效。要绝对确保这一点，你可以在MySQL监视器中使用一条FLUSH PRIVILEGES命令来重新载入授权表。

### 2.8.3 移除权限

移除权限和添加权限一样简单，只不过是使用REVOKE命令，而不是使用GRANT命令。REVOKE命令的语法如下。

```
REVOKE privileges
ON databasename.tablename
FROM username@hostname;
```

我们授权许可来使用INSERT命令，采用同样的方式，我们也可以通过使用DELETE命令从mysql数据库的表中删除记录，从而取消前面的授权许可。然而，这需要你熟悉字段和表，并且使用REVOKE命令会更容易和安全。

要把用户john向myCompany数据库中的INSERT能力收回，可以使用如下一条REVOKE语句。

REVOKE INSERT
---------------

```
ON myCompany.*  
FROM john@hostname
```

对权限表中的数据修改会立刻生效，但是，为了让服务器立刻意识到你的修改，在MySQL监视器中可以使用FLUSH PRIVILEGES命令。

## 2.9 小结

在Windows和Mac OS X上安装MySQL是一个简单的过程，因为有一个基于向导的安装方法。Linux/UNIX用户没有一个基于向导的安装过程，但是，按照一组简单的命令来拆包MySQL客户机和服务器二进制文件也并不困难。Linux/UNIX用户也可以使用RPM来安装MySQL。

安全性总是首要的问题，可以采取几个步骤来确保安全地安装MySQL。即便你不是服务器管理员，还是应该能够认识到安全漏洞并且提醒服务器管理员注意。

MySQL服务器不应该作为root用户运行。此外，在MySQL中创建用户时总是应该有一个密码，并且他们获取的权限也应该是定义好的。

对于每一次请求，MySQL都是在两个步骤的过程中使用权限表，从而知道你是谁以及你从哪里连接而来，并且信息的每一部分必须和权限表中的一个条目相匹配。此外，所使用的用户身份必须有执行你所做请求的类型的具体权限。

可以使用GRANT命令来添加用户权限，它使用一个简单的语法来向mysql数据库中的user表添加条目。REVOKE命令通常也很简单，用来移除这些权限。

## 2.10 Q&A

**Q:** 如何完全删除一个用户？**REVOKE**命令只是删除了权限。

**A:** 要完全从权限表中删除一个用户，必须对mysql数据库的user表使用一个专门的**DELETE**命令。

**Q:** 如果我告诉我的**Internet**服务提供商停止把**MySQL**当作**root**进程运行，而他却不听，该怎么办？

**A:** 换一家**Internet**服务提供商。如果你的**Internet**服务提供商没有认识到像把数据库这样重要的东西当作**root**用户进程运行的风险，并且不听取你的要求，就另外找一家提供商。有些提供商只需每月\$2.95的费用（甚至是免费），他们不会将重要进程作为**root**用户进程运行。

## 2.11 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 判断真假：对于从一个远程主机到MySQL的安全连接，SSH是完全可接受的方法。
2. 每次向MySQL服务器发出一个请求的时候，MySQL检查哪3部分信息？
3. 要授予localhost上的名为bill的用户对于BillDB数据库的所有表的SELECT、INSERT和UPDATE权限，使用什么命令？另外，为了安全性的考虑，推荐的这条语句中漏掉了哪些信息？

## 解答

1. 真。SSH将主机之间的数据加密，因此，确保了到服务器的连接是安全的。

2. 你是谁，你从哪里访问，允许你执行什么操作。

3. 命令是

```
GRANT SELECT, INSERT, UPDATE  
ON BillDB.*  
TO bill@localhost;
```

遗漏的重要信息是用户的密码。



## 思考题

1. 考虑一下你希望在表级限制访问命令的情况。例如，你不希望实习级别的管理员拥有关闭公司数据库的权限。
2. 如果你在MySQL中有管理员权限，使用几条GRANT命令创建一些假的用户。你所指定的表和数据库实际上是否存在无关紧要。
3. 使用REVOKE来移除在思考题2中所创建的用户的一些权限。

## 第3章 安装和配置Apache

在本章中，你将学习：

- 如何安装**Apache**服务器。
- 如何对**Apache**做配置修改。
- **Apache**的日志和配置文件存储在何处。

在本章中，我们将安装Apache Web服务器并熟悉它的主要组件，包括日志和配置文件。

## 3.1 Apache的当前版本及未来版本

当你访问位于<http://httpd.apache.org>的Apache HTTPD服务器站点，将会看到关于Apache 2.0.x、Apache 2.2.x和Apache 2.4.x版本的发布声明。正如根据版本号可以猜到的，Apache 2.0.x是这三个版本中最老的版本。Apache软件基金会（Apache Software Foundation）维护了所有这3个版本，但是，Apache 2.4.x的功能包含了对过滤、缓存、负载均衡以及其他系统功能很好的支持。这也是本章中使用的版本。然而，无论你选择安装Apache 2.2.x还是Apache 2.0.x，本书中所有的PHP和MySQL代码都能够像描述的那样工作。实际上，你将会发现，相当多的托管提供商仍然使用服务器的Apache 2.0.x版本，而不是Apache 2.2.x，更不要说最新的Apache 2.4.x版本了。如果你在按照本书介绍的内容安装Apache 2.4.x的时候有任何问题，请尝试安装其前一个版本（例如，Apache 2.2.x）；安装说明非常相似。本章的安装说明针对Apache HTTPD服务器的2.4.1版，这是在编写本书的时候可获得的该软件的最好版本。

Apache软件基金会（Apache Software Foundation）对包含增强安全或问题修复的升级使用次版本号。次版本没有预定的发布计划，当对代码进行了扩展和修复并且进行了彻底的测试后，Apache软件基金会就会用一个新的次版本号来发布一个新的版本。

当你购买本书时，有可能次版本号已经发生改变，变为2.4.1或更高。如果是这种情况，你应该阅读改变的列表，从<http://httpd.apache.org/download.cgi>的下载区可以找到这个列表的链接，其中所有关于安装或配置过程的改变组成了变更列表的大部分内容。

虽然在次版本升级中不太可能所有的安装说明都改变，但是应该养成始终检查自己安装和维护的软件的改变日志的习惯。如果你阅读本书期间次版本发生了改变，但是在改变日志中没有新的安装改变记录，你只要用心记下新的版本号，并且当版本号需要出现在安装说明和相关的图中的时候，用最新版本号替换就行了。

## 3.2 选择合适的安装方法

当安装过程进行到适当的位置，并获得一个基本的Apache安装时，你有几个选择。Apache是开源软件，这意味着你有权使用软件的全部源代码，也就是允许你编译自己自定义的服务器。另外，预先编译好的Apache二进制代码发布对大多数现代UNIX平台都可用。最后，Apache已经捆绑到很多种Linux发布上，你还可以从软件供应商那里购买带有支持包的商业版。如果使用的是Linux/UNIX，本章中的示例将教你如何从源代码安装Apache；如果你计划在Windows系统上运行Apache，本章中的示例将教你如何使用安装程序。

### 3.2.1 从源代码安装

从源代码安装将带给你最大的灵活性，因为它允许你安装一个自定义服务器，移除不需要的模块，并且用第三方模块扩展服务器。从源代码安装Apache使你轻松升级到最新版本并快速应用安全补丁，而由供应商提供的最新版本可能要数天甚至数周后才能发布。对于简单安装，从源代码安装Apache的过程并不是特别难，但是当包括第三方模块和库时，安装过程的复杂度将有所增加。

### 3.2.2 安装一个二进制代码版本

Linux/UNIX二进制安装版本可以从供应商获得，也可以从Apache软件基金会网站下载。针对只有有限的系统管理知识的用户或者没有特别配置需要的用户，他们提供了一种简便的安装Apache的方法。第三方商业供应商提供一个增加了应用程序服务器、附加的模块、支持等预打

包的Apache安装版本。Apache软件基金会为Windows系统提供了一个安装程序,Windows平台中的编译器并不如Linux/UNIX中的编译器常用。

## 3.3 在Linux/UNIX上安装Apache

这部分介绍了如何在Linux/UNIX上安装Apache 2.4.1的最新版本。成功地从源代码安装Apache的一般步骤如下。

1. 下载软件。
2. 运行配置脚本。
3. 编译代码并安装它。

下面的部分将详细介绍这些步骤。

### 3.3.1 下载Apache源代码

官方的Apache下载站点是<http://httpd.apache.org/download.cgi>。你可以找到Apache源代码的几个版本，它们是用不同压缩方法打包的。发布文件首先用tar应用程序打包，然后用gzip应用程序或compress应用程序压缩。如果你已经在自己的系统上安装了gunzip应用程序，那么下载\*.tar.gz版本。这个应用程序在诸如FreeBSD和Linux的开放源代码操作系统中已经默认安装。如果你的系统上没有安装gunzip，那么下载\*.tar.Z文件（gunzip并没有包含在多数商业UNIX操作系统的默认安装中）。

你想下载的文件将命名为类似httpd-VERSION.tar.gz的文件，其中，VERSION是Apache的最新发布版本。例如，把Apache 2.2.8下载并存成一个名为httpd-2.4.1.tar.gz的文件。把下载文件放到为源文件保留的目录，例如/usr/src/或/usr/local/src/。



### 3.3.2 解压源代码

如果下载了用gzip压缩的tarball（带有tar.gz后缀），可以用gunzip应用程序（gzip发布的一部分）来解压缩。

提示：

tarball是对使用tar应用程序压缩软件的一个通用别名。

你可以通过输入如下命令来解压缩并拆包软件。

```
# gunzip < httpd-2.4*.tar.gz | tar xvf -
```

解压缩tarball创建了一个目录结构，带有名为httpd *VERSION* 的顶级目录。把当前目录切换到这个顶级目录，准备配置软件。

### 3.3.3 准备编译Apache

你可以通过使用在顶级发布目录中的configure脚本，指定生成的二进制代码中将有特性。默认情况下，把Apache编译为一系列静态编译的标准模块，并安装在/usr/local/apache2目录。如果乐于使用这些设置，可以输入如下命令来配置Apache。

```
# ./configure
```

然而，为给第4章中PHP安装做准备，你需要确保把mod\_so模块编译进Apache。以UNIX共享对象(\*.so)格式命名的这个模块，可以让Apache激活诸如PHP这样的动态模块。在特定位置（在本例中是/usr/local/apache2/）配置Apache来安装它自己，并允许使用mod\_so，输入如下命令。

```
# ./configure --prefix=/usr/local/apache2 --enable-so
```

configure脚本的目的是解决关于查找库、编译时选项、特定平台差异等所有的事情，并创建一系列叫做makefiles的专用文件。makefiles包含执行不同任务（例如安装Apache）的指令，叫做targets。这些文件由make应用程序来读取，它将执行这些任务。如果一切顺利，执行configure后你将看到一系列和刚才执行的不同检查相关的消息，并将返回到命令提示行。

```
...
configure ok

creating test/Makefile
config.status: creating docs/conf/httpd.conf
...
config.status: executing default commands
#
```

如果configure脚本失败，将显示警告，提醒你追查刚刚安装的其他软件，例如编译器或库。在安装所有缺少的软件并从顶级目录删除config.log和config.status文件后，你可以再次尝试configure命令。

如果配置过程最终给出一个警告，显示你没有安装APR，那么，访问<http://apr.apache.org/>并且下载APR和APR-util软件包，并且将它们解压缩到你的httpd-VERSION源目录的srclib子目录中。一旦安装了它们，再次运行配置命令。类似的，如果配置过程最终给出一个警告，说你没有安装PCRE，访问<http://www.pcre.org>并下载该文件，根据该Web站点的说明在你的系统上安装它。安装完成后，再次运行配置命令。

和Apache 2.2.x的安装过程不同，在Apache 2.4.x的安装过程中，这两项需求都会有所改变。

### 3.3.4 编译和安装Apache

`make`应用程序读取保存在`makefiles`中的信息并编译服务器和模块。在命令行输入`make`命令以编译Apache。你将看到指示编译的进程的几条消息，最后将回到命令提示行。编译结束后，你可以通过在命令提示行中输入`make install`来安装Apache。`makefiles`将安装文件和目录，并返回到命令提示行。

```
...
Installing header files
Installing build system files
Installing man pages and online manual
...
make[1]: Leaving directory '/usr/local/bin/httpd-2.4.1'
#
```

正如`configure`命令中的`--prefix`开关指定的那样，Apache发布文件现在将在`/usr/local/apache2`目录中。要测试`httpd`二进制代码是否已经正确编译，在命令提示行输入如下命令。

```
# /usr/local/apache2/bin/httpd -v
```

你将看到如下输出（你的版本和编译日期将有所不同）。

```
Server version: Apache/2.4.1 (Unix)
Server built:   March 12 2012 11:47:22
```

除非你想学习如何在Mac OS X 或Windows上安装Apache，否则请直接跳到“Apache配置文件结构”一节学习Apache的配置文件。

## 3.4 在Mac OS X上安装Apache

你很幸运，Apache已经安装到Mac OS X中。默认情况下，Apache服务器二进制代码位于/usr/sbin/httpd。诸如httpd.conf等配置文件，即Apache的主要配置文件，保存在/etc/httpd。因为Apache已经安装并且完全准备好使用PHP，请直接跳到“Apache配置文件结构”部分学习Apache配置文件以及如何使用它。

### 注意：

如果你想要使用针对Mac OS X的一次性软件包，可以像第1章中所介绍的那样使用XAMPP，或者你可以安装来自<http://www.mamp.info>的MAMP软件包。

## 3.5 在Windows上安装Apache

### 注意：

在编写本书时，Apache 2.4.x对于Windows来说还不可用。因此，如下的说明是针对2.2.x的。当Apache 2.4.x变得可用的时候，其安装过程将会是类似的。

Apache 2.2可以在绝大多数Windows平台上运行，并且提供比Apache 2.0和Apache 1.3版本更佳的性能和稳定性。你可以从源代码编译Apache，但是因为不是多数Windows用户都有编译器，本节用MSI安装程序版本进行安装。

在安装Apache之前，你大概想确定当前在自己的机器上是否已经运行了一个Web服务器（例如，Apache的前一个版本、Microsoft Internet Information Server或Microsoft Personal Web Server），可能希望卸载或者禁用已存在的服务器。你可以同时运行几个Web服务器，但是它们必须运行在不同的地址和端口组合上。

在下载安装程序之前，花一点时间（这是一个非常重要的时刻）在下载页面（位于<http://httpd.apache.org/download.cgi>）查找一条句子，内容是“If you are downloading the Win32 distribution, please read these important notes.”。指向这些要点的URL是<http://www.apache.org/dist/httpd/binaries/win32/README.html>。

Apache软件基金会维护这个页面是为了那些想运行Apache服务器版本的Windows用户。在这个页面里几乎有每一个目前仍在使用的

Windows版本的要点，同样地，你最好提起兴趣阅读一下这里的信息。如果你正在运行Apache，无论作为一个产品还是开发服务器，我保证你在这个要点页面找到相关的信息。

当你已经准备好安装时，查找名为Win32 Binary（MSI安装程序）的链接。当下载完安装程序后，双击这个文件开始安装过程。你将看到一个欢迎界面，如图3-1所示。

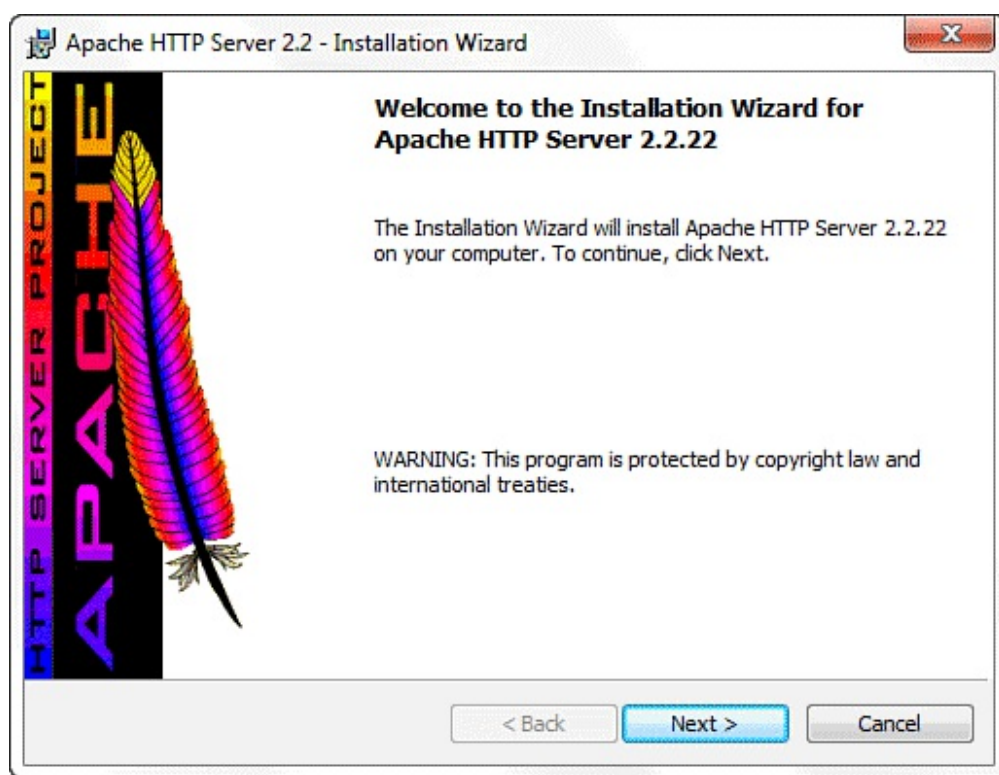


图3-1 Windows安装程序欢迎界面

点击Next按钮继续安装过程，将提示你接受Apache授权许可。授权许可主要表示：除了声明是你编写的这个软件之外，你可以用这个软件做任何想做的事——包括做私人的修改。但是一定要阅读这个授权许可，以便你完全理解这些条款。

接受授权许可之后，安装程序给出一个Apache的简短介绍。紧接着，它要求提供你的计算机的基本信息，如图3-2所示。这包括服务器的完整的网络地址（举例来说，mycomputer.mydomain.com）和管理员的Email地址。服务器名是客户端将用来访问服务器的名字，而管理员的E-mail地址将被添加到错误信息中，以便于在出现错误时访问者知道如何联系你。

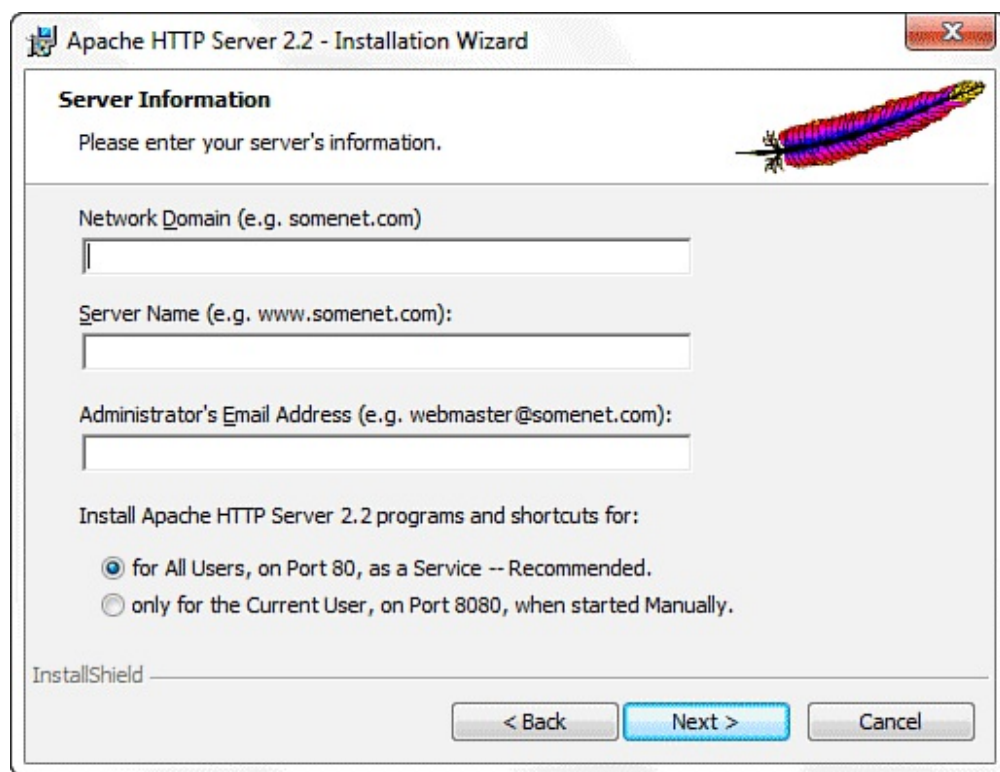


图3-2 基本信息界面

这个界面还将提示选择安装哪种启动方式——是将Apache作为服务启动还是手动启动Apache。把Apache作为服务安装，Apache将在Windows每次启动时自动运行，并且你可以通过标准Windows服务管理工具控制它。为当前用户安装Apache，需要手动启动Apache并且设置默认端口，Apache在这个端口上监听对8080（而不是80）的请求。选择合



适的单选按钮并点击Next按钮继续。

**提示：**

如果你的机器没有完整的网络地址，使用localhost或127.0.0.1作为服务器名。

下一个界面让你选择安装的类型，典型（typical）或自定义（custom）。典型安装（Typical installation）意味着将安装Apache二进制代码和文档，但是标头和库将不安装。这是供选择的最佳选项，除非你准备编译自己的模块。

自定义安装（Custom installation）让你选择是否安装标头文件或文档。选择目标安装目录后，默认是C:\Program Files\Apache Software Foundation\Apache 2.2，程序将处理安装过程。如果一切顺利，它将显示一个如图3-3所示的最终界面。

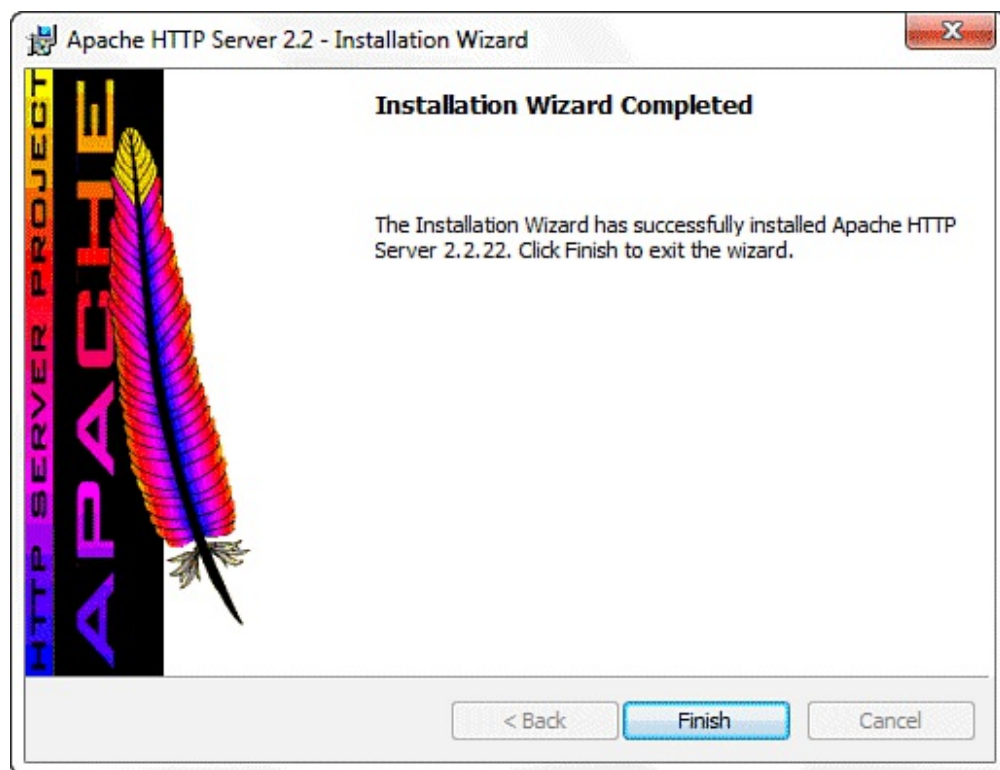


图3-3 成功安装界面

在下一节里，你将学习有关Apache配置文件的内容，并最终启动你的新服务器。

## 3.6 Apache配置文件结构

Apache把它所有的配置信息都保存到文本文件中。主文件叫做httpd.conf。这个文件包含指令和容器，它们允许定制Apache安装。指令(Directive)配置Apache的特定设置，例如授权、性能和网络参数。容器(Container)指定这些设置作用的上下文。例如，授权配置可以作用于作为一个整体的服务器、一个目录或单个文件。

### 3.6.1 指令

下面的规则适用于Apache指令语法。

- 指令参数跟在指令名后面。
- 指令参数用空格分隔开。
- 参数的数量和类型根据指令的不同而不同，某些指令没有参数。
- 一个指令占用单独的一行，但是通过在指令前一行的结尾加一个反斜杠字符(\)，可以在另一行继续这个指令。
- 井号符(#)要在指令之前，且必须每行一个。

#### Apache服务器文档

在<http://httpd.apache.org/docs/2.2/mod/quickreference.html> 提供了一个关于指令的快速参考。虽然稍后将学习一些基本的指令，但是你应该使用联机文档补充自己的知识。

指令的Apache文档通常都遵循如下格式。

- **Description** ——这个条目提供了指令的一个简单说明。

- **Syntax** ——这个条目解释了指令选项的格式。必需的参数以斜体显示，可选参数以斜体显示并用括号括起来。
- **Default** ——如果指令有一个默认值，将在这里显示它。
- **Context** ——这个条目详细介绍了可以显示指令的容器或区域。容器将在下一小节介绍。可能的值是server config、virtual host、directory和.htaccess。
- **override** ——Apache的指令属于不同的种类。Override字段用于指定哪个指令种类可以显示在.htaccess per-directory配置文件中。
- **Status** ——这个条目指明指令是否在Apache(core)中编译，是否属于绑定模块之一（base或extension，取决于默认情况下是否编译它们），是多进程模块（Multi-Processing Module, MPM)的一部分，还是用Apache绑定但不准备用在产品服务器（experimental）中。
- **Module** ——这个条目指明指令属于哪个模块。
- **Compatibility** ——这个条目包含有关支持这个指令的Apache版本的信息。

文档中的这些项目可以找到指令更深入的解释，并且有关指令或文档的参考资料可能显示在最后。

### 3.6.2 容器

指令容器，也叫做段（section），限制指令作用的作用域。指令如果没有在一个容器内，则属于默认服务器作用域（server config），并且作为一个整体作用于服务器。

如下是Apache默认的指令容器。

- **<VirtualHost>** ——VirtualHost指令指定一个虚拟服务器。Apache允许你用单个安装的Apache运行不同的Web站点。这个容器中的指令作用于一个特定的Web站点。这个指令接受域名或IP地址以及一个可选的端口作为参数。你将在第29章学习关于虚拟主机更多的内容。
- **<Directory>, <DirectoryMatch>** ——这些容器允许指令作用于某一目录或者文件系统中的目录组。Directory容器获取一个目录或目录结构参数。容器包括的指令作用于指定目录以及它们的子目录。DirectoryMatch容器允许指定正则表达式结构作为参数。举例来说，下列内容允许www目录的所有二级子目录的一个匹配，它由4个数字组成，例如一个按年和月（0212代表2012年2月）命名的目录。

```
<DirectoryMatch "^/www/.*/[0-9]{4}">
```

- **<Location>, <LocationMatch>** ——这些容器允许指令作用于某个请求的URL或URL结构。它们与<Directory>, <DirectoryMatch>类似。LocationMatch获取一个正则表达式作为参数。例如，下列内容匹配包含“/my/data”或“/your/data”的目录。

```
<LocationMatch "/(my|your)/data">
```

- **<Files>, <FilesMatch>** ——类似于Directory和Location的容器，Files段允许目录作用于某个文件或文件结构。

容器包含指令，如程序清单3.1所示。

---

```
1: <Directory "/some/directory">
2: SomeDirective1
3: SomeDirective2
4: </Directory>
5: <Location "/downloads/*.html">
6: SomeDirective3
7: </Location>
8: <Files "\.(gif|jpg)">
9: SomeDirective4
10: </Files>
```

---

示例指令*SomeDirective1*和*SomeDirective2*将作用于/some/directory目录及其子目录。指令*SomeDirective3*将作用于指向带有.html扩展名的页面的URL，这个页面位于/downloads/URL下。*SomeDirective4*将作用于带有.gif或.jpg扩展名的所有文件。

### 3.6.3 条件评估

Apache提供对条件容器的支持。只有当某些条件符合时，才执行这些容器中的封装指令（directives enclosed）。

- **<IfDefine>** ——如果把一个指定的命令行开关传递给Apache的可执行程序，将执行这个容器内的指令。只有当把-DMyModule开关传递给将要执行的Apache二进制代码时，才会执行程序清单3.2中的指令。可以直接在命令行传递这个开关，也可以通过修改apachectl脚本来传递这个开关，稍后将在本章的“Apache相关命令”一节介绍。

IfDefine容器也接受求反的参数。也就是说，只有非-DMyModule参数作为命令行参数传递时，<IfDefine !MyModule>段中的指令（注意叹号在MyModule名字之前）才执行。

- **<IfModule>** ——只有当作为参数传递的模块出现在Web服务器中时，IfModule段中的指令才会执行。举例来说，Apache载入一个对不同MPM提供支持的默认httpd.conf配置文件。正如我们在程序清单3.3中见到的那样，只有属于编译到Apache中的MPM的配置才会执行。这个示例的目的是阐明将只执行指令组中的一个指令。

程序清单3.2 IfDefine示例

---

```
1: <IfDefine MyModule>
2: LoadModule my_module modules/libmymodule.so
3: </IfDefine>
```

---

程序清单3.3 IfModule示例

---

```
1: <IfModule prefork.c>
2: StartServers      5
3: MinSpareServers   5
4: MaxSpareServers   10
5: MaxClients        20
6: MaxRequestsPerChild 0
7: </IfModule>
8:
9: <IfModule worker.c>
10: StartServers      3
11: MaxClients        8
12: MinSpareThreads   5
13: MaxSpareThreads   10
14: ThreadsPerChild   25
15: MaxRequestsPerChild 0
16: </IfModule>
```

---

### 3.6.4 ServerRoot指令

ServerRoot指令获取一个单个的参数：一个指向服务器作用目录的目录路径。其他指令中的所有相对路径引用都是相对于ServerRoot的值。如果你在Linux/UNIX上从源代码编译Apache，如本章前面所述，ServerRoot的默认值是/usr/local/apache2。对于Mac OS X用户，



ServerRoot的默认值是/Library/WebServer。如果使用的是Windows安装程序，ServerRoot的默认值是C:\Program Files\Apache Software Foundation\Apache 2.2\。

### 3.6.5 per-directory配置文件

Apache使用per-directory配置文件来允许指令在主配置文件http.conf之外存在。这些专用文件可以存放到文件系统中。如果请求一个文档，而该文档保存在包含上述专用文件之一的一个目录中或者它下面的任何子目录中，Apache将处理这些专用文件的内容。所有适用的per-directory配置文件的内容将会合并处理。例如，如果Apache接收到一个对/usr/local/apache2/htdocs/index.html文件的请求，它将在/、/usr、/usr/local、usr/local/apache2和/usr/local/apache2/htdocs目录中，按照顺序查找per-directory配置文件。

#### 注意：

启用per-directory配置文件会有性能损失。Apache必须执行大量磁盘操作来查找每个请求中的这些文件，即使这些文件不存在。

默认把per-directory配置文件称为.htaccess，这样叫法是有历史原因的。它们最初用于保护对包含HTML文件的目录的访问。

AccessFileName指令允许你把per-directory配置文件名从.htaccess改为其他的名字。它接受一个文件名列表，当查找per-directory配置文件时Apache将使用该列表。

要确定一条指令是否可以在per-directory配置文件中覆盖，检查指

令语法定义的Context:字段是否包含.htaccess。Apache指令属于不同的组，就像在指令语法说明中的Override字段中指定的那样。Override字段可能的值如下。

- **AuthConfig** ——授权指令。
- **FileInfo** ——控制文档类型的指令。
- **Indexes** ——控制目录索引的指令。
- **Limit** ——控制主机访问的指令。
- **Options** ——控制指定目录特性的指令。

通过使用AllowOverride指令，可以控制哪个指令组能够显示在per-directory配置文件中。AllowOverride也能获取一个All或None参数。All意味着属于所有组的指令都能显示在配置文件中。None意味着在一个目录以及它的所有子目录中禁用per-directory配置文件。程序清单3.4展示了如何作为一个整体对服务器禁用per-directory配置文件。这提升了性能，也是默认的Apache配置。

程序清单3.4 禁止per-directory配置文件

---

```
1: <Directory />
2: AllowOverride none
3: </Directory>
```

---

## 3.7 Apache日志文件

默认情况下，Apache包含两个日志文件。`access_log`文件用于跟踪客户请求；`error_log`文件用于记录重要事件，例如错误或者服务器重新启动。这些文件从你第一次启动Apache时就存在了，这些文件在Windows平台下名为`access.log`和`error.log`。

### 3.7.1 `access_log`文件

当客户从服务器请求一个文件时，Apache记录与这个请求相关的参数，包括客户的IP地址、请求的文档、HTTP状态码和当前时间。程序清单3.5显示了示例日志文件条目。第26章将介绍如何修改已记录的参数。

程序清单3.5 `access_log`条目示例

---

```
1: 127.0.0.1 - - [12/Mar/2012:08:33:25 -0700] "GET / HTTP/1.1" 200 44
2: 127.0.0.1 - - [12/Mar/2012:08:33:25 -0700] "GET /favicon.ico HTTP/1.1" 404 209
```

---

### 3.7.2 `error_log`文件

`error_log`文件包含错误信息、启动信息和服务器生命周期中的其他重大事件。当你使用Apache遇到问题时，这是应该查看的第一个地方。程序清单3.6展示了`error_log`条目示例。

程序清单3.6 `error_log`条目示例

---

```
1: Starting the Apache2.4 service [The Apache2.4 service is running.]
2: Apache/2.4.1 (Unix) configured -- resuming normal operations
3: [Tue Mar 13 08:29:34 2012] [notice] Server built: Mar 12 2012 11:47:22
4: [Tue Mar 13 08:29:34 2012] [notice] Parent: Created child process 3504
5: [Tue Mar 13 08:29:35 2012] [notice] Child 3504: Child process is running
6: [Tue Mar 13 08:29:35 2012] [notice] Child 3504: Acquired the start mutex.
```

---

### 3.7.3 其他文件

httpd.pid文件包含运行Apache服务器的进程ID号。可以手动使用这个号码发送信号给Apache，接下来一节将详细介绍它。scoreboard文件是Linux/UNIX Apache上的当前配置文件，基于进程的MPM用这个文件和它们的子进程通信。一般来说，不需要考虑这些文件。

## 3.8 Apache相关命令

Apache发布包含几个可执行程序。这一节只介绍服务器二进制程序和相关脚本。第25章和第29章将介绍Apache发布包含的其他应用程序。

### 3.8.1 Apache服务器二进制程序

Apache可执行程序在Linux/UNIX和Mac OS X中叫做httpd，在Windows中叫做apache.exe。它接受几个命令行选项，参见表3-1中的说明。你可以通过在Linux/UNIX上输入`/usr/local/apache2/bin/httpd -h`，在Mac OS X上输入`/usr/sbin/httpd -h`，或在Windows上从命令行提示输入`apache.exe -h`，来获取一个完整的选项列表。

表3-1 httpd选项

选 项	意 义
-D	允许传递一个可以用于<IfDefine>段处理的参数
-l	列出编译进服务器的模块
-v	显示版本号和服务器编译时间
-f	如果和编译时默认值不同，允许传递http.conf的位置

Apache运行后，可以在Linux/UNIX和Mac OS X上使用kill命令来发

送信号给Apache父进程。信号提供一种发送命令给进程的机制。要发送一个信号，执行下列命令。

```
# kill -SIGNAL pid
```

在这个语法中，*pid*是进程ID号，*SIGNAL*是下列内容之一。

- **HUP** ——停止服务器。
- **USR1**或**WINCH** ——温和地重新启动；使用哪个信号取决于在什么操作系统下。
- **SIGHUP** ——重新启动。

如果对配置文件做了某些变化并且想让它们立即生效，你必须发信号通知Apache配置文件已经发生变化。通过停止并启动服务器或发送一个重新启动的信号，你可以做到这一点。这告诉Apache重新读取它的配置文件。

普通的重新启动会导致服务的瞬间暂停。温和的重新启动采取了一种不同的方法：服务于一个客户的各个线程或进程将继续执行当前请求，但当它结束时，它将被终止，并用一个采用新配置的新线程或新进程代替它。这允许Web服务器实现没有停机时间的无缝操作。

在Windows上，你可以使用apache.exe可执行程序发信号给Apache。

- **apache.exe -k restart** ——告诉Apache重新启动。
- **apache.exe -k graceful** ——告诉Apache温和地重新启动。
- **apache.exe -k stop** ——告诉Apache停止。

你可以在开始菜单项中访问Apache安装程序创建的这些命令的快捷方式。如果把Apache作为一项服务安装，可以通过使用Windows服务界面启动或停止Apache：在控制面板中选择管理工具并点击服务图标即可。

### 3.8.2 Apache控制脚本

尽管在Linux/UNIX上使用httpd二进制代码控制Apache是可能的，但是推荐使用apachectl工具。apachectl支持程序把通用功能包装到一个易于使用的脚本中。要使用apachectl，输入如下命令。

```
# /usr/local/apache2/bin/apachectl command
```

在这个语法中，command可以是stop、start、restart或graceful。你也可以编辑apachectl脚本的内容以添加额外的命令行选项。某些OS发布提供另外的脚本来控制Apache，请检查发布中包含的文档。

## 3.9 第一次启动Apache

在启动Apache之前，你应该检查最基本的信息设置已经在Apache配置文件httpd.conf中完成了。下面一节将介绍配置Apache并启动服务器所需的基本信息。

### 3.9.1 检查你的配置文件

你可以用自己喜欢的文本编辑器编辑Apache httpd.conf文件。在Linux/UNIX和Mac OS X中，它可能是vi或emacs。在Windows中，可以用记事本或写字板。必须记住把配置文件存为纯文本格式，这是Apache唯一能够识别的格式。

第一次启动Apache只有两个参数需要修改：服务器的名字以及它要监听的地址和端口。服务器的名字是当Apache需要引用它自己时使用的一个参数,例如重定向请求的时候。

Apache通常可以从机器的IP地址算出它的服务器名，但这不是绝对的。如果服务器没有一个有效的DNS条目，你可能需要指定机器的IP地址中的一个。如果服务器没有连接到网络（你可能想在一台独立的机器上测试Apache），你可以使用值127.0.0.1，这是一个回送地址

（loopback address），默认端口号是80。如果已经有一个服务器运行在机器的80端口上，你可能需要改变这个值；或者如果你在Linux/UNIX和Mac OS X系统上没有管理员权限，只有root用户可以绑定到保留端口（那些小于1024的端口），那可能也需要改变这个端口号。



你可以用Listen指令改变监听地址和端口值。Listen指令获取一个端口号或用冒号分隔的一个IP地址和一个端口。如果只指定了一个端口，Apache将在机器中所有可用的IP地址上监听对那个端口的请求。如果还提供了一个IP地址，Apache将只监听那个地址和端口的组合。例如，Listen 80告诉Apache在所有IP地址上监听对80端口的请求。Listen 10.0.0.1:443告诉Apache只监听10.0.0.1上的443端口。

ServerName指令允许你定义服务器名字，服务器将把这个名字报告给所有自引用的URL。这个指令接受一个DNS名字和一个可选的端口，两者用冒号分隔开。确定ServerName有一个有效值。否则，服务器将不能正常运行，例如产生错误的重定向。

在Linux/UNIX和Mac OS X平台上，你可以使用User和Group指令来指定服务器将作为哪个用户和组ID运行。nobody用户对于大多数平台而言是一个好的选择。然而，在HP-UX平台用这个用户ID会出现问题，所以必须创建并使用一个不同的用户ID，例如www。

### 3.9.2 启动Apache

要在Linux/UNIX上启动Apache，切换到包含apachectl脚本的目录并执行下面的命令。

```
# /usr/local/apache2/bin/apachectl start
```

Mac OS X用户可以在命令行提示下输入如下命令。

```
# /usr/sbin/httpd
```

要在Windows上手动启动Apache，在开始菜单中的Apache HTTP Server 2.2程序组中，在Control Apache Server目录下点击Start Apache in

Console链接。如果把Apache作为服务安装，必须启动Apache服务。

如果一切顺利，你可以使用浏览器访问Apache。显示一个默认安装页面，如图3-4所示。如果不能启动Web服务器或显示一个错误页面，请参阅后续的“故障排除”一节。确定你在Listen指令中指定的端口之一上访问Apache——通常是80或8080端口。

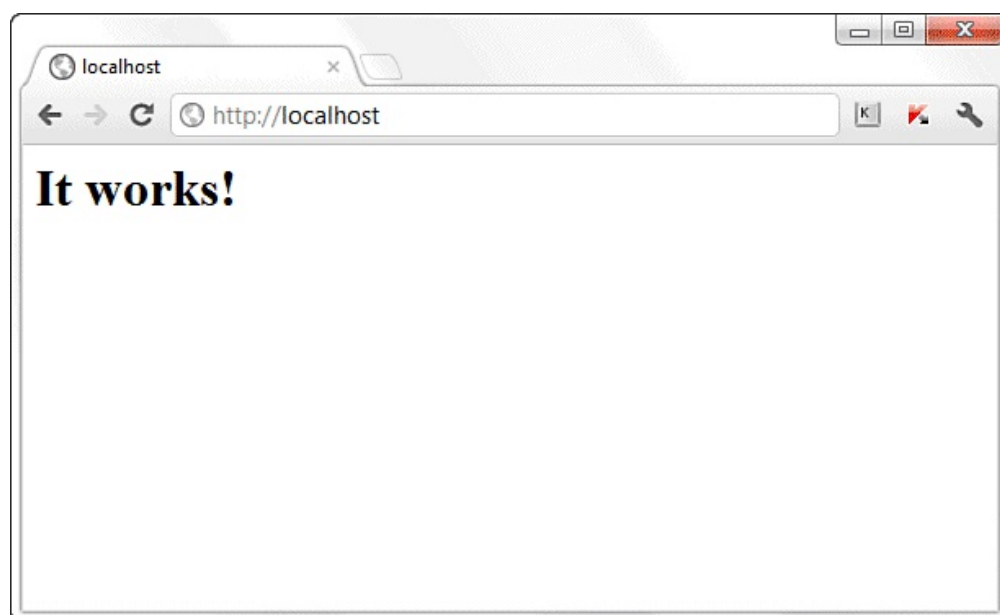


图3-4 已经安装了Apache

## 3.10 故障排除

下面的小节介绍了第一次启动Apache时可能遇到的几个常见问题。

### 3.10.1 已有Web服务器

如果机器上已经运行了一个服务器并监听同样的IP地址和端口的组合，Apache将不能成功启动。你将在错误日志文件中看到如下的一个条目，它表明Apache不能绑定到端口。

```
[crit] (48)Address already in use: make_sock: could not bind...[alert] no listening sockets available, shutting down
```

要解决这个问题，需要停止运行的服务器或改变Apache配置以监听一个不同的端口。

### 3.10.2 不允许绑定到端口

如果没有管理员权限并试图绑定到一个保留端口（在0到1024之间），你将得到一个如下的错误信息。

```
[crit] (13)Permission denied: make_sock: could not bind to address 10.0.0.2:80  
[alert] no listening sockets available, shutting down
```

要解决这个问题，就必须在启动Apache之前作为管理员登录或改变端口号，8080是一个经常使用的非保留端口。

### 3.10.3 拒绝访问

如果没有权限读取配置文件或写入日志文件，你可能不能启动Apache且将会得到一个如下的错误信息。

(13)Permission denied: httpd: could not open error log file

如果一个用户试图运行Apache，而Apache由另一个不同的用户编译并安装，将可能产生这个问题。

### 3.10.4 错误组设置

你可以配置Apache在某个用户名和组下运行。对于运行中的服务器用户名和组，Apache有默认值。有时默认值并不正确，你将得到一个包含setgid: unable to set group id的错误。

要在Linux/UNIX和Mac OS X上解决这个问题，必须在配置文件中把Group指令的值修改为一个正确的值，为已有的组检查/etc/groups文件。

## 3.11 小结

本章介绍了在Linux/UNIX、Mac OS X和Windows上安装并运行Apache 2.2服务器的不同方法。其中包括了二进制代码和源代码安装，并介绍了基本的编译时选项。另外，我们学习了服务器配置文件的位置和用于修改Apache配置的命令的语法。我们学习了两种主要的日志文件：access\_log和error\_log。最后，我们学习了在Linux/UNIX、Mac OS X和Windows平台上如何使用Apache控制脚本或Apache服务器二进制代码启动和停止服务器。

## 3.12 Q&A

**Q:** 我如何能够启动一个清除性的编译？

**A:** 如果你需要从源代码编译一个新Apache并且不想早期编译的结果影响新的编译结果，最好的办法是运行make clean命令。它将仔细清除所有已有二进制程序、中间对象文件等。

**Q:** per-directory配置文件有什么用？

**A:** 尽管per-directory配置文件对服务器性能有影响，但它们对委托管理有用。因为每次发生请求时都读取per-directory配置文件，所以对配置做改变时不需重新启动服务器。

你可以允许Web站点的用户不必拥有管理员权限就能自行修改配置。这样，他们可以分部分地对他们的主页进行密码保护。

**Q:** 一个正确的ServerName指令对你意味着什么？

**A:** DNS系统用于将IP地址和域名关联起来。当服务器生成一个URL时返回ServerName的值。如果要使用某一域名，你必须确定它包含在DNS系统中并且访问站点的客户可以使用它。

### 3.13 实践练习

实践练习是设计用来帮助你预料可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 练习题

1. 你如何才能指定安装Apache的位置？
2. <Location>段和<Directory>段之间主要的区别是什么？
3. 重新启动和温和地重新启动之间有什么区别？



## 解答

1. Linux/UNIX用户可以使用configure脚本的--prefix选项指定安装Apache的位置。如果在这个位置已经有一个安装，配置文件将保留，但是二进制文件将被替换。在Windows中，可以在安装向导中设置这个位置。
2. Directory段作用于文件系统对象；Location段作用于Web页面的地址栏中的元素。
3. 在正常重新启动期间，服务器停止，然后启动，这将导致一些请求丢失。温和地重新启动允许Apache子进程/线程继续为当前的请求服务，直到运行新配置的子进程/线程能够取代它们。

## 思考题

1. 练习不同类型的服务器停止运行和重新启动过程。
2. 做一些配置修改，例如不同端口分配和ServerName变化。

## 第4章 安装和配置PHP

在本章中，你将学到：

- 如何安装**PHP**。
- 如何测试**PHP**的安装。
- 出错的时候如何寻求帮助。
- 基本的**PHP**语言。

在本章中，你将会获取、安装和配置**PHP**，并且对**Apache**的安装做出一些基本的改变。

## 4.1 PHP的当前版本和未来版本

本章中的安装说明针对PHP 5.4.0，这是该软件的当前版本。

PHP Group使用修订和小版本的发布来更新所包含的安全性扩展或bug的修复。这些发布并不遵从一套发布时间规划，当扩展或修复添加到代码中并经过彻底的测试之后，PHP Group就使用新的修订号发布一个新的版本。

当你购买本书的时候，小版本可能已经变化到5.4.1或更高版本。如果是这种情况，你应该阅读位于<http://www.php.net/ChangeLog-5.php> 的变化列表，来了解关于安装/配置过程变化的信息。这些过程是本章的主要内容。

尽管在两个次版本更新之间不可能所有的安装过程都要变化，但你还是应该养成习惯查看自己所安装和维护的软件的更新日志。如果你阅读本书的时候，确实出现了一个次版本的变化，但更新日志中并没有提到安装的变化，你只需要用心记下，并且当出现在安装说明和相应的图中的时候，用新的版本号替代就行了。

## 4.2 在带有Apache的Linux/UNIX上编译PHP

在本节中，我们将看到在带有Apache的Linux/UNIX上安装PHP的过程。这个过程对于任何类似UNIX的操作系统来说多少都有些相同。尽管你可能能够为自己的系统找到PHP的预编译的版本，但是从源代码编译PHP还是给了你对于构建到自己的二进制文件中的功能的较大控制权。

要下载PHP发布文件，到PHP的主页<http://www.php.net/>，并且找到Downloads部分。找到最新版本的源代码，例如，我们使用的是5.4.0。你的发布的名字将会类似php-VERSION.tar.gz，其中VERSION是最近的发布版本号。这个文档将会是一个压缩的tar文件，因此，我们需要解压它。

```
# gunzip < php-VERSION.tar.gz | tar xvf -
```

把下载文件放到为源文件保留的一个目录中，如/usr/src/或/usr/local/src/。当你的发布解压后，应该切换到PHP发布目录。

```
# cd php-VERSION
```

在你的发布目录下，可以找到一个名为configure的脚本。当从命令行运行configure脚本的时候，这个脚本会接受所提供的额外信息。这些命令行参数控制着PHP将要支持的功能。在这个例子中，我们将会包含在Apache和MySQL的支持下安装PHP所需的基本选项。稍后，我们还将讨论一些可用的configure选项，在本书中，它们都是相关的。

```
# ./configure --prefix=/usr/local/php \  
--with-mysql=/usr/local/mysql/bin/mysql_config \  
--with-apxs2=/usr/local/apache2/bin/apxs
```

如果你将MySQL或Apache安装到了与这里的配置所给出的路径不同的位置，那么，确保在命令中用相应的目录路径进行替换。

脚本运行之后，会返回到命令提示行。

```
...  
Generating files  
updating cache ./config.cache  
creating ./config.status  
creating php5.spec  
creating main/build-defs.h  
creating scripts/phpize  
creating scripts/man1/phpize.1  
creating scripts/php-config  
creating scripts/man1/php-config.1  
creating sapi/cli/php.1  
creating main/php_config.h  
creating main/internal_functions.c  
creating main/internal_functions_cli.c  
+-----+  
| License: |  
| This software is subject to the PHP License, available in this |  
| distribution in the file LICENSE. By continuing this installation |  
| process, you are bound by the terms of this license agreement. |  
| If you do not agree with the terms of this license, you must abort |  
| the installation process at this point. |  
+-----+  
  
Thank you for using PHP.  
  
#
```

从命令提示行执行一条make命令，接着执行make install命令。这些命令将会完成PHP的编译和安装过程，并且返回命令提示行。

```

...
chmod 755 /usr/local/apache2/modules/libphp5.so
[activating module 'php5' in /usr/local/apache2/conf/httpd.conf]
Installing PHP CLI binary:      /usr/local/php/bin/
Installing PHP CLI man page:    /usr/local/php/php/man/man1/
Installing PHP CGI binary:      /usr/local/php/bin/
Installing build environment:   /usr/local/php/lib/php/build/
Installing header files:        /usr/local/php/include/php/
Installing helper programs:     /usr/local/php/bin/
    program: phpize
    program: php-config
Installing man pages:           /usr/local/php/php/man/man1/
    page: phpize.1
    page: php-config.1
...
#

```

你还需要确保两个非常重要的文件复制到正确的位置。首先，使用如下命令来把php.ini的推荐版本复制到其默认位置。稍后我们将在本章中学习有关php.ini的更多知识。

```
# cp php.ini-development /usr/local/lib/php.ini
```

接下来，如果安装过程没有把PHP共享对象文件复制到Apache安装目录中的正确位置（通常安装程序会复制PHP共享对象文件，从make install命令的输出可以看到），那你需要执行如下命令。

```
# cp libs/libphp5.so /usr/local/apache2/modules/
```

你现在应该可以配置和运行Apache了，但是，在直接开始“在Linux/UNIX上整合PHP和Apache”这一节之前，让我们先介绍一些额外的配置选项。

### 4.2.1 额外的Linux/UNIX配置选项

在前面的一节中，当运行PHP configure脚本的时候，我们可以包含一些命令行参数来确定PHP引擎将要包含的那些功能。configure脚本本

身给出了一个可用选项的列表，包括我们所使用过的那个列表。从PHP发布的目录，输入如下命令。

```
# ./configure --help
```

这条命令产生一个长长的列表，以便你将其存入到文件并且在空闲的时候阅读它。

```
# ./configure --help > configoptions.txt
```

如果在PHP安装之后发现还有额外的功能想要添加到PHP中，只需要再次运行配置和编译过程。这么做将会生成一个新版本的libphp5.so，并且将其放置到Apache的目录结构中。你所需要做的只是重新启动Apache以载入新文件。

## 4.2.2 在Linux/UNIX上集成PHP和Apache

要确保PHP和Apache能够协同工作，我们需要检查httpd.conf配置文件并且潜在地向其中添加一些项目。首先，看看如下的一行内容。

```
LoadModule php5_module          modules/libphp5.so
```

如果这行内容没有出现，或者在这行的开头出现了一个“#”号，你必须添加一行或者删除这个“#”号。这一行告诉Apache使用PHP编译过程所创建的PHP共享对象文件(libphp5.so)。

接下来，查找如下内容。

```
#  
# AddType allows you to add to or override the MIME configuration  
# file mime.types for specific file types.  
#
```



在这一部分内容后面添加如下一行。

```
AddType application/x-httpd-php .php
```

这条语句确保了PHP引擎将会解析以.php和.html扩展名结尾的文件。你所选择的文件名可能有所不同。

保存这个文件，然后重新启动Apache。当你查看自己的error\_log的时候，应该会看到如下的一行：

```
[Tue Mar 13 10:42:47 2012] [notice] Apache/2.4.1 (Unix) PHP/5.4.0 configured
```

PHP现在已经是Apache Web服务器的一部分了。如果你想要了解如何在Mac OS X平台上安装PHP，请继续阅读。否则，可以跳到“测试安装”一节。

## 4.3 在Mac OS X上安装PHP

在带有Apache的Mac OS X上安装PHP有几个不同的选项，包括前面小节所介绍的从源代码编译。一些用户可能会发现，最简单的方法是从一个预编译的二进制包来安装PHP，例如，MacPorts (在<http://www.macports.org/>)，它是一次性安装包XAMPP的一部分，或者MAMP (在<http://www.mamp.info>)。然而，如果你习惯使用命令行，我建议你按照4.2节的说明来进行。

## 4.4 在Windows上安装PHP

在Windows上安装PHP也只需要下载发布文件而已。要下载PHP发布文件，请到PHP的主页<http://www.php.net/>，按照链接找到Downloads页面。找到线程安全的ZIP包的最新版本，例如，我们使用的是5.4.0。你下载的发布的名字将会类似于php-VERSION.zip，其中VERSION是最近的发布版本号。

文件下载之后，双击它启动解压缩软件。发布打包时带有的相关路径名已经存在了，因此，把文件解压缩到名为C:\php\的一个新目录，所有的文件和子目录都将放置到这个新目录下面。

接着，访问C:\php\目录并把文件php.ini-recommended复制为php.ini。为了获得和Apache一起使用的PHP的基本版本，我们需要对Apache配置文件做一些细微的修改。

### 注意：

在一些Windows系统上，你可能需要设置一个明确的环境变量，以便让PHP正确地运行；即便你不确定这个变量是否是必须的，设置它也不会引起什么危害，因此，没有理由不这么做。要了解将PHP目录添加到PATH环境变量的更多信息，参见位于<http://www.php.net/manual/en/faq.installation.php#faq.installation.addtopath>的PHP FAQ的相关条目。

## 在Windows上集成PHP和Apache

要确保PHP和Apache能够协同工作，我们需要对httpd.conf配置文件

添加一些项目。首先，找到httpd.conf配置文件如下的部分。

```
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule access_module modules/mod_access.so
...
#LoadModule vhost_alias_module modules/mod_vhost_alias.so
```

在这部分的末尾，添加如下内容。

```
LoadModule php5_module C:/php/php5apache2_2.dll
```

此外，添加如下的内容以确保Apache知道php.ini的位置。

```
PHPIniDir "C:/php/"
```

接下来，找到如下的部分。

```
#
# AddType allows you to add to or override the MIME configuration
# file mime.types for specific file types.
#
```

在这部分的末尾添加下面的一行。

```
AddType application/x-httpd-php .php
```

这条语句确保了PHP引擎可以解析以.php和.html扩展名为结尾的文件。你对文件名的选择可能有所不同。

保存httpd.conf文件，然后重新启动Apache。服务器启动后应该没有警告，PHP现在已经是Apache Web服务器的一部分了。

## 4.5 php.ini基础

当你编译或安装了PHP之后，仍然可以使用php.ini文件来改变其行为。在Linux/UNIX系统上，这个文件的默认位置是/usr/local/php/lib，或者是在配置时所使用的PHP安装位置的lib子目录。在Windows系统中，这个文件应该在PHP目录中，或者在Apache httpd.conf文件中的PHPIniDir值所指定的另外一个目录中。

php.ini文件中的指令有两种格式：值和标记。值指令的格式是一个指令名以及一个等号隔开的一个值。可能的值对于不同的指令来说各有不同。标记指令的格式是一个指令名以及一个等号隔开的一个正的或负的项。正的项包括1、On、Yes和True；负的项包括0、Off、No和False；空白忽略。

### 注意：

在Windows系统上，重要的是为extension\_dir指令提供正确的值。如果你将PHP安装在C:\php位置，那么，extension\_dir的值应该是“C:\php\ext”。

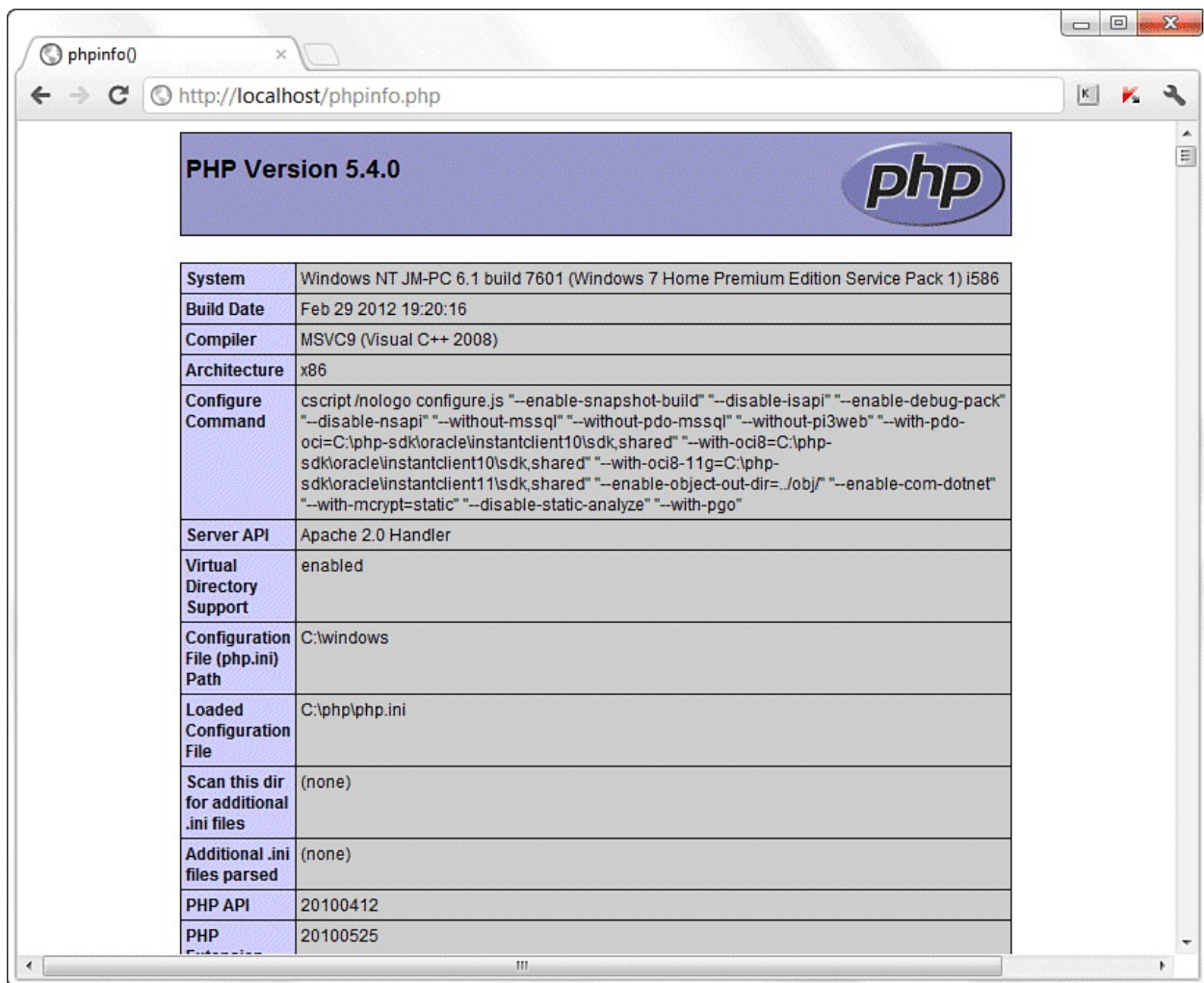
你可以随时改变php.ini的设置，但是改变之后，需要重新启动服务器以便让改变生效。在这里，不妨花点时间阅读一下自己的php.ini文件，看看可以配置的项目有哪些。

## 4.6 测试安装

测试PHP安装的最简单的方法就是使用phpinfo()函数创建一个小的测试脚本。这个函数将会产生一个长长的配置信息列表。打开文本编辑器并输入如下的命令行。

```
<?php phpinfo(); ?>
```

把这个文件保存为phpinfo.php，并且将其放置到Web服务器的文档根目录下，即Apache安装的htdocs子目录或者Mac OS X上的/Library/WebServer/Documents。使用Web浏览器访问该文件，你应该会看到如图4-1所示的内容。




<div> <div>PHP Version 5.4.0</div>  </div>	
System	Windows NT JM-PC 6.1 build 7601 (Windows 7 Home Premium Edition Service Pack 1) i586
Build Date	Feb 29 2012 19:20:16
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack"           "--disable-nsapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared"           "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet"           "--with-mcrypt=static" "--disable-static-analyze" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\windows
Loaded Configuration File	C:\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20100412
PHP	20100525

图4-1 phpinfo()的结果

phpinfo()具体的输出取决于你的操作系统、PHP版本以及配置选项。

## 4.7 获取安装帮助

通过Internet获取帮助总是很方便，尤其是对于涉及到开源软件的问题。然而，在单击发送按钮之前，不妨稍等片刻。不管你的安装、配置或编程问题看上去如何难以解决，但肯定不会只有你才遇到这种情况。可能有人已经解决了你的问题。

当我们遇到困难，首先应该求助的资源就是PHP官方网站<http://www.php.net/>（尤其是使用说明手册<http://www.php.net/manual/>）。如果在这里没有找到答案，别忘了PHP站点是可以搜索的。你所寻求的参考说明有可能隐藏在一个发布消息中或者在一个常见问题解答文件中。你也可以在<http://www.php.net/search.php> 搜索邮件列表文档。这些文档凝聚了PHP社区的很多智慧，并且提供了非常多的信息资源。不妨花点时间尝试几个搜索关键词的组合。

如果确定自己的问题没有被解决过，可以向PHP社区提出问题并得到服务。你可以从<http://www.php.net/mailling-lists.php> 加入到PHP邮件列表。尽管这些列表常常具有很大的信息容量，但你还是可以从中学到很多。如果你专门从事PHP脚本编程，至少应该订阅一个摘要列表，当你订阅了和自己相关的列表，可以考虑发布你的问题。

当你发布一个问题的时候，包含尽可能多的信息是个不错的想法，但不要像写小说似的长篇累牍。通常需要提供如下的相关信息。

- 你的操作系统类型。
- 你所运行或安装的PHP版本。



- 你的配置选项。
- 在安装失败之前的`configure`或`make`命令的任何输出。
- 引发问题的代码的一个相对完整的示例。

为什么所有这些都和在一个邮件列表张贴问题有关呢？首先，培养研究技能会对你大有裨益。一个好的研究者通常能够快速而高效地解决问题。在一个技术列表中张贴一个幼稚的问题，往往会需要等待而得到的只是一条消息或者两个引用而已，它们只能够让你知道应该首先从哪里去查找问题的答案。

其次，别忘了在一个邮件列表并不像一个技术支持呼叫中心。没有人会因为回答你的问题而得到报酬。尽管如此，你还是会看到一个令人难忘的智慧和知识的宝库，其中包括一些PHP的创造者的思想。一个好的问题及其解答将会被存档，以帮助其他的程序员。多次询问已经解答的问题则只会增加更多的噪音。

尽管如此，不要担心把问题张贴到邮件列表中。PHP开发者是一群文明而乐于助人的人，而且通过让问题引起社区的注意，你可能也帮助其他人解决了同样的问题。

## 4.8 PHP脚本基础

让我们直接跳到PHP脚本。首先，打开你所喜爱的文本编辑器，像HTML文档一样，PHP文件也是由纯文本构成的。你可以使用任何文本编辑器来创建它们，并且，大多数流行的HTML编辑器和编程IDE（集成开发环境）都支持PHP。

寻找PHP友好的编辑器的一个不错的站点是[www.php-editors.com](http://www.php-editors.com)。输入程序清单4.1中的例子并将文件保存到Web服务器的根目录，将其命名为first.php。

程序清单4.1 一个简单的PHP脚本

---

```
1: <?php
2:     echo "<h1>Hello Web!</h1>";
3: ?>
```

---

如果不能在为PHP脚本提供运行服务的机器上直接工作，则可能需要一个FTP或SCP客户端把保存的文档上传到服务器。当文档位于服务器上的适当位置时，你就可以使用浏览器来访问它了。如果一切正常，将会看到脚本的输出。图4-2显示了脚本first.php的输出。



图4-2 成功:脚本first.php的输出

4.8.1 开始和结束一个PHP语句块

在编写PHP的时候，你需要通知PHP引擎你想执行的是哪些命令。如果不能通知PHP引擎这些命令，所编写的代码将会被误认为HTML并且直接输出到浏览器。你可以把自己的代码设计为带有专门标记的PHP，这些标记表示了PHP代码块的开始和结束。表4-1给出了4种PHP分隔标记。

表4-1 PHP开始和结束标记

标 记 类 型	开 始 标 记	结 束 标 记
标准标记	<?php	?>
短标记	<?	?>
ASP式标记	<%	%>

脚本标记	<script language="php">	</script>
------	-------------------------	-----------

在表4-1所示的标记中，只有标准标记和脚本标记保证对任何配置都有效。短标记和ASP式标记必须在php.ini中显式地启用。

要激活对短标记的识别，必须确保在php.ini中将short\_open\_tag设置为On。

```
short_open_tag = On;
```

要激活对ASP式标记的识别，必须确保在php.ini中将asp\_tags设置为On。

```
asp_tags = On;
```

**提示：**

要确保可移植的、可复用的代码，最好使用标准标记而不是短标记或ASP式标记。原因很简单，服务器的配置总是唯一的，使用标准标记是因为你知道在任何配置上都可以使用它。

在编辑了php.ini并且重新启动Apache之后，就可以在脚本中使用四种标记类型中的任何一种了。这在很大程度上与个人偏好有关，当然如果你有意要在自己的脚本中包含XML，你应该避免使用短标记(<? ?>)而使用标准标记(<?php ?>)。

**注意：**

字符序列<?告诉一个XML解析器将有一个处理指令，因此该序列经常包含在XML文档中。如果你在脚本中包含XML并且短标记可用，PHP引擎就可能混淆XML处理指令和PHP开始标记。如果你有意在文档中加入XML，那么就关闭短标记。

让我们给出程序清单4.1中的代码的一些合法写法。你可以使用已经见到的4个PHP开始和结束标记中的任何一个。

```
<?
echo "Hello Web!";
?>

<?php
echo "Hello Web!";
?>

<%
echo "Hello Web!";
%>

<script language="php">
echo "Hello Web!";
</script>
```

你也可以把单行代码放入到与PHP开始和结束标记相同的一行中，示例如下。

```
<?php echo "Hello Web!"; ?>
```

现在，你知道如何定义一段PHP代码，下面让我们进一步看看程序清单4-1本身的含义。

## 4.8.2 echo语句和print()函数

简单地说，echo语句用来输出数据。在大多数情况下，echo的任何输出最终在浏览器中都是可见的，也可以使用print()函数来替代echo语句。使用echo或print()只是个人习惯的问题，当你查看其他人的脚本时，两种用法都会见到。

再来看看到目前为止我们所见到过的代码，注意程序清单4-1中以

分号结束的唯一那一行代码。这个分号告诉PHP引擎，一条语句结束了，并且，这是目前为止我们所学到的关于编码语法的最重要的一点。

语句（statement）表示对PHP引擎的一条指令。更广泛地讲，对于PHP来说，它就像是用英语写或说的一个句子。一个英语句子通常应该用一个句点结束，一条PHP语句通常应该以一个分号结束。这条规则的例外之一是包含了其他语句的语句，以及结束一个代码块的语句。然而，在大多数情况下，没有用分号结束一条语句将会引起PHP引擎的混淆并且产生一个错误。

### 4.8.3 组合HTML和PHP

程序清单4.1中的脚本是纯PHP的。只需要简单地把HTML加到PHP开始和结束标记的外围，就能够把这个脚本加入到一个HTML文档中，如程序清单4.2所示。

程序清单4.2 加入到HTML中的一个PHP脚本

---

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>A PHP script including HTML</title>
5: </head>
6: <body>
7: <h1><?php echo "hello world"; ?></h1>
8: </body>
9: </html>
```

---

正如你所见到的，把PHP代码加入到一个主控HTML文档中只是代码输入上的小问题。PHP引擎忽略PHP开始和结束标记外围的所有内容。如果你把程序清单4-2中的内容保存为helloworld.php，并将其放置到文档根目录，然后使用浏览器查看它，如图4-3所示，你可以看到粗

体的hello world。如果你要查看文档源文件，如图4-4所示，清单看上去确实像一个普通的HTML文档。

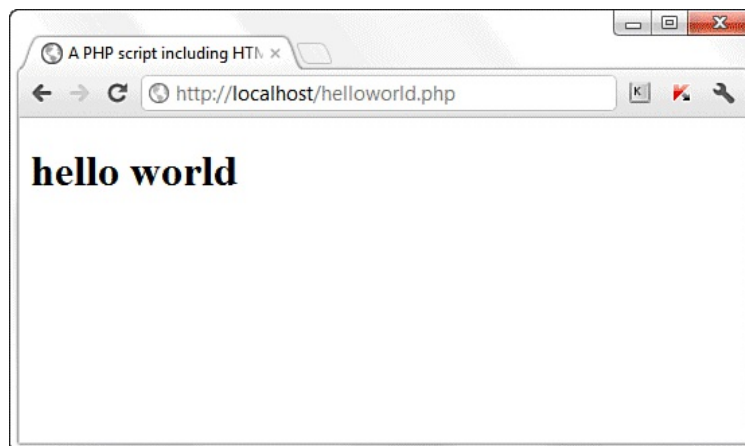


图4-3 helloworld.php的输出在浏览器中的样子

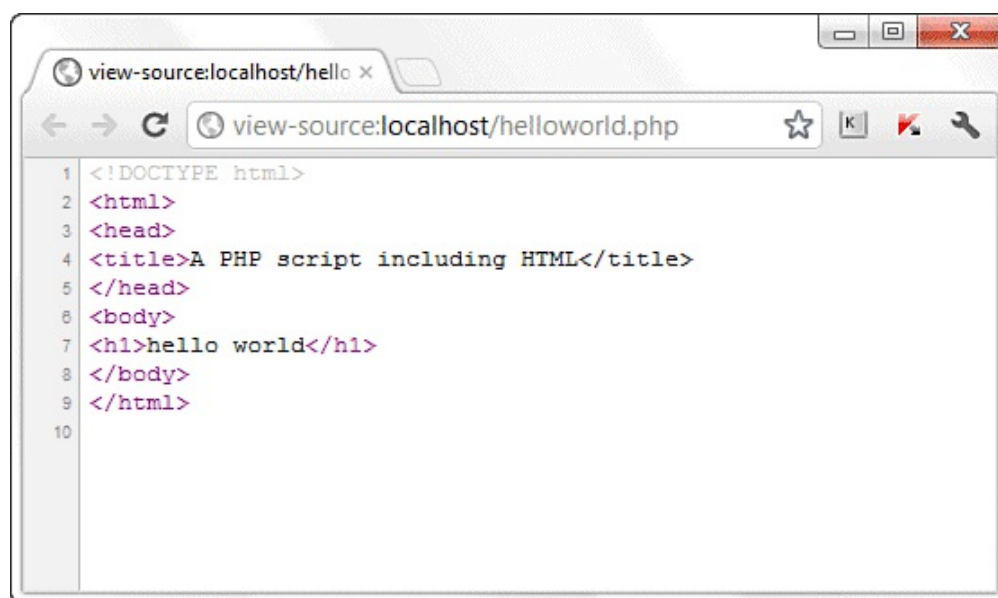


图4-4 helloworld.php的输出的HTML源代码

只要你愿意，你可以在一个单个的文档中包含任意多个PHP代码块，这些代码块可以根据需要散布在HTML中。尽管可以在一个单个的文档中有多个代码块，但可以将它们看做一个单个的脚本。定义在第一

个代码块中的任何变量，通常也可以供后续的代码块使用。

#### 4.8.4 为PHP代码添加注释

那些在编写的时候看上去干干净净的代码，当你6个月后试图改进它的时候，看上去是那么的杂乱无章。在编写代码的时候为代码添加注释，以后将会节省你的时间，并且使得其他的程序员更容易使用你的代码。

注释（comment）就是脚本中会被PHP引擎忽略的文本。注释可以使得代码更具可读性，或者用来说明一个脚本。

单行注释以两个斜杠开始（//），这是首选的方式，也可以以一个斜杠或一个井号开始。PHP引擎忽略这些符号到行末或者这些符号到PHP结束标记之间的所有文本。

```
// this is a comment
#  this is another comment
```

多行注释以一个斜杠后面跟着一个星号（/\*）开始，结束的时候是一个星号后面跟着一个斜杠（\*/）。

```
/*
this is a comment
none of this will
be parsed by the
PHP engine
*/
```



## 4.9 小结

在本章中，我们学习了如何在Linux/UNIX、Mac OS X或Windows上安装和配置PHP 5.4.0，以便和Apache一起使用。学习了在Linux/UNIX编译脚本中的各种configure选项，可以修改它们来改变所支持的功能。还学习了有关php.ini的知识，以及如何改变其指令的值。

使用phpinfo()函数，我们可以测试安装并且产生其配置值的一个列表。我们使用一个文本编辑器创建了一个简单的PHP脚本。我们介绍了可以用来开始和结束PHP代码块的4种标记。

最后，我们学习了如何使用echo语句或print()函数向浏览器发送数据，并且把HTML和PHP脚本组合到同一个文件中。在下一章，我们将使用这些技巧来测试PHP语言的一些基本组成部分，包括变量、数据类型和操作符。

## 4.10 Q&A

**Q:** 我们已经介绍了**Linux/UNIX**、**Mac OS X**、**Windows**以及**Apache Web**服务器的**PHP**安装。这是否意味着本书介绍的内容不适用于我的服务器和操作系统？

**A:** 不是的。**PHP**的一个最大优点就是它可以在多种平台上运行。你可以在**PHP**手册中找到针对不同**Web**服务器和数据库支持的不同配置指令的安装说明。尽管本书中的例子都是特别针对**PHP**、**MySQL**和**Apache**的，但在使用不同的**Web**服务器或数据库的时候，只需要略作修改就可以使用这些例子。

**Q:** 最好的开始和结束标记是什么？

**A:** 这在很大程度上是个人偏好的问题。但为了可移植性，标准标记(`<?php ?>`)是最佳的选择。

**Q:** 在编写**PHP**代码的时候，应该避免使用哪些编辑器？

**A:** 不要使用为了打印而格式化文本的字处理软件（如**Microsoft Word**）。即便使用这种软件以纯文本的格式保存你所创建的文件，隐藏的字符也可能遍布代码之中。

**Q:** 我该在何时注释我的代码？

**A:** 这又是一个个人偏好的问题。某些较短的脚本具有很好的自解释性，即便在很长一段时间之后也是如此。不管脚本的长度和复杂性如何，都应该注释你的代码。从长远来讲，注释代码常常会节省你的时间，减少挫折感。

## 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 从Linux/UNIX操作系统，你如何获得有关配置选项（你传递给PHP发布中的configure脚本的选项）的帮助？
2. 为了确保能够识别.php扩展名的文件，你应该在Apache配置文件中添加一行什么？
3. PHP的配置文件叫什么？
4. 一个人浏览你的Web站点可以读取已经成功安装的PHP脚本的源代码吗？

## 解答

1. 可以通过在PHP发布目录下调用configure脚本并把--help参数传递给它，从而获取有关配置选项的帮助信息，命令如下。

```
./configure --help
```

2. 确保Apache能够把.php扩展名的文件当作PHP脚本的一行命令如下。

```
AddType application/x-httpd-php .php
```

3. PHP的配置文件叫做php.ini。

4. 不能，这个用户只能看到脚本的输出。

## 思考题

1. 在你的系统上安装PHP。如果已经安装了，查看php.ini文件并检查你的配置。

2. 熟悉创建、上传和运行PHP脚本的过程。特别是创建你自己的“hello world”脚本。为它添加HTML代码，并且添加其他的PHP语句块。使用不同的PHP分隔符标记来实验。你的配置支持哪一个标记？看一下php.ini文件以证实你的发现。别忘了为你的代码添加一些注释。

## 第2部分 PHP语言结构

第5章 PHP的组成部分

第6章 PHP的流程控制功能

第7章 使用函数

第8章 使用数组

第9章 使用对象



## 第5章 PHP的组成部分

在本章中，你将学到：

- 变量——它们是什么、为什么我们需要使用它们，以及如何使用它们。
- 如何定义和访问变量。
- 数据类型。
- 一些很常用的操作符。
- 如何使用操作符来编写表达式。
- 如何定义和使用常量。

在本章中，我们将亲身了解一些PHP脚本语言的具体细节。新接触编程的读者可能会觉得不知所措，不过不用担心，你以后可以随时回过头来参考本章。请把注意力放在对概念的理解上，而不是记住所介绍的功能上，因为这些元素会在本书的脚本中重复出现。如果第一次没有掌握它们，你最终还是会掌握它们的。

如果你已经是一个有经验的程序员，可以略过本章，因为它介绍了一些和全局变量、数据类型以及改变类型相关的、PHP特有的功能。

## 5.1 变量

变量（variable）是我们可以定义的一个特殊的容器，随后它可以“容纳”一个值，例如，一个数字、字符串、对象、数组或者一个布尔值。变量是编程的基本概念，没有变量，对脚本中用到的每个特定值都需要进行直接编码。如下的直接编码语句把两个数字加在一起并且显示出结果，这解决了一个简单的数学问题。

```
echo (2 + 4);
```

然而，这段代码只对那些确实想知道2加4的和是多少的人才有用。为了超越这个限制，我们可以编写一段代码来得出其他两组数的和，例如3和5。然而，这种编程方法很荒谬，而这正是变量的用武之地。

变量允许我们为运算创建模板，例如，把两个数相加，而不用担心变量具体所表示的值。当脚本运行的时候，值将会赋给变量，可能通过用户输入、通过一个数据库查询或者通过脚本中之前的另一个操作的结果获得值。换句话说，当脚本中的数据很可能改变的时候（不管是在脚本的生存期内，还是当它传递给稍后使用的另一个脚本时），应该使用变量。

一个变量由你所选取的一个名字以及前面的一个美元符号（\$）组成。变量名可以包含字母、数字以及下划线（\_），但不能包含空格。变量名必须以一个字母或一个下划线开始。下面是一些合法的变量。

```
$a;  
$a_longish_variable_name;  
$2453;  
$sleepyZZZZ;
```

#### 提示：

变量名应该有意义而且要风格一致。例如，如果你的脚本要处理名字（name）和密码（password），不要为名字创建一个名为\$*n*的变量，为密码创建一个\$*p*变量，因为对于除了你以外的任何人，这都是没有意义的名字。而且如果你数周之后再回顾这段脚本，你可能会把\$*n*当作是“number”而不是“name”的变量，或认为\$*p*代表“page”而不是“password”。如果一个合作者想要修改你的脚本该怎么办？他怎么知道\$*n*和\$*p*代表什么？你可以对自己的脚本中的变量使用任何命名惯例，只要名字是具有描述性的并且遵从其他人可以理解的某种模式。

一个分号（`;`）（又叫做指令终止符，`instruction terminator`）用来结束一条PHP语句。上述的代码段中的分号并非变量名的一部分，而是用来结束声明变量的语句。要声明一个变量，你只需将其包含到自己的脚本中。当你声明一个变量的时候，通常会在同一条语句中为它赋一个初始值，如下所示。

```
$num1 = 8;
```

```
$num2 = 23;
```

上述代码行声明了两个变量，并且使用赋值操作符（`=`）来为它们赋值。我们将会在本章后面的5.3节更详细地了解赋值。在为变量赋值之后，我们可以将变量看作变量值本身一样。换句话说。

```
echo $num1;
```

等于

```
echo 8;
```

只要将值8赋给了\$num1。

### 5.1.1 全局变量

变量除了命名变量的规则以外，还有一些有关可用性的规则。通常，赋给一个变量的值只在它所在的函数或脚本中有效。例如，如果你有一个scriptA.php保存了一个名为\$name、值为joe的变量，如果想要创建一个scriptB.php，它也使用\$name变量，我们可以给第二个\$name变量赋值jane，而毫不影响scriptA.php中的变量。对于每个脚本，\$name变量的值都是局部的（local），并且赋给的值相互之间是独立的。

然而，我们也可以在脚本或者函数中把\$name变量定义为全局的（global）。如果在scriptA.php和scriptB.php中，将\$name变量定义为一个全局变量，并且，这两个脚本彼此连接（也就是说，一个脚本调用另一个脚本，或者包含另一个脚本），那么对于共享的\$name变量将只有一个值。全局变量作用域的例子将在第7章详细说明。

### 5.1.2 超全局变量

除了自己创建的全局变量，PHP还有几个叫做超全局变量（superglobal）的预定义的变量。这些变量总是存在的，并且它们的值也总是对所有的脚本可用的。如下的每个超全局变量，实际上都是其他变量的一个数组。

- \$\_GET包含了通过GET方法提供给一个脚本的任何变量。
- \$\_POST包含了通过POST方法提供给一个脚本的任何变量。
- \$\_COOKIE包含了通过cookie提供给一个脚本的任何变量。
- \$\_FILES包含了通过文件上传提供给一个脚本的任何变量。
- \$\_SERVER包含了像标头、文件路径和脚本位置等信息。
- \$\_ENV包含了作为服务器环境的一部分提交给一个脚本的任何变量。

- `$_REQUEST`包含了通过GET、POST或COOKIE输入机制提供给一个脚本的任何变量。
- `$_SESSION`包含了在一个会话中当前注册的任何变量。

本书中的例子将会在可能的情况下使用超全局变量。在脚本中使用超全局变量对于创建安全的应用程序很重要，因为超全局变量减少了用户注入式攻击进入到脚本的可能性。通过编码，可以让脚本只接受你想要的内容，并且按照你定义的方式（例如，从使用POST方法的一个表单或从一个会话）来接受，可以消除一些由于松散地编写脚本而引发的问题。

## 5.2 数据类型

不同的数据类型占用不同的内存量，并且在一个脚本中操作它们的时候可能区别对待它们。一些编程语言因此要求程序员提前声明一个变量所要包含的数据的类型。相反，PHP是类型宽松的语言，这意味着它将在数据被赋给每个变量的时候才确定数据类型。

这种自动类型真是祸福相依。一方面，它意味着变量可以灵活地使用，例如，一个变量可以存储字符串，并且随后它可以在脚本中存储整数或某些其他数据类型。另一方面，在较大的脚本中，如果你特别希望一个变量存储某种数据类型而它所存储的东西完全不同的话，这种灵活性可能会导致问题。例如，假设你在编写用来操作一个数组变量的代码，如果所讨论的变量是一个数字值，而不是数组值，当代码试图在这个变量上执行特定于数组的操作的时候，就会发生错误。

表5-1给出了PHP中可用的8种标准的数据类型。

表5-1 标准数据类型

类 型	例 子	说 明
Boolean	true	特定值true或false中的一个
Integer	5	一个整数
Float或Double	3.234	一个浮点数

String	"hello"	字符的一个集合
Object		类的一个实例
Array		键和值的一个有序的集合
Resource		对一个第三方资源（如数据库）的引用
NULL		一个未初始化的变量

**Resource**类型经常由处理外部应用程序或文件的函数返回。**NULL**类型是为了那些已经声明但还没有赋值的变量保留的。

**PHP**有几个函数可以测试一个变量的特定类型的合法性，实际上，每个类型都有一个函数。这是一组**is\_\***函数，例如**is\_bool()**测试一个给定的值是否是布尔值。程序清单5.1把几个不同的数据类型分配给单个的变量，然后使用相应的**is\_\***函数来测试这个变量。代码中的注释显示脚本处理到哪里了。

**提示：**

关于调用函数，我们将在第7章了解更多信息。

程序清单5.1 测试一个变量的类型

---

```
1: <?php
2: $testing; // declare without assigning
3: echo "is null? ".is_null($testing); // checks if null
4: echo "<br/>";
5: $testing = 5;
6: echo "is an integer? ".is_int($testing); // checks if integer
7: echo "<br/>";
8: $testing = "five";
9: echo "is a string? ".is_string($testing); // checks if string
10: echo "<br/>";
11: $testing = 5.024;
12: echo "is a double? ".is_double($testing); // checks if double
13: echo "<br/>";
14: $testing = true;
15: echo "is boolean? ".is_bool($testing); // checks if boolean
16: echo "<br/>";
17: $testing = array('apple', 'orange', 'pear');
18: echo "is an array? ".is_array($testing); // checks if array
19: echo "<br/>";
20: echo "is numeric? ".is_numeric($testing); // checks if is numeric
21: echo "<br/>";
22: echo "is a resource? ".is_resource($testing); // checks if is a resource
23: echo "<br/>";
24: echo "is an array? ".is_array($testing); // checks if is an array
25: echo "<br/>";
26: ?>
```

---

把上述代码保存到一个名为testtype.php的文本文件中，并且把这个文件放入到你的Web服务器文档根目录下。当你通过Web浏览器访问这个脚本的时候，会产生如下的输出。

```
is null? 1
is an integer? 1
is a string? 1
is a double? 1
is boolean? 1
is an array? 1
is numeric?
is a resource?
is an array? 1
```

我们在第2行声明了\$testing变量，却没有为它赋值，因此，当我们在第3行测试这个变量看它是否为空的时候（使用is\_null()），结果是



1(true)。

**注意：**

如果你已经配置了PHP以显示所有的注意、警告和错误，当你运行如下这段脚本的时候，将会看到如下的一条注意事项。

**Notice:** Undefined variable: `testing` in `/path/to/testtype.php` on line **3**

当你使用`php-development.ini`而不是`php-production.ini`的时候，注意会默认地打开。调试脚本的时候，打开注意很有用。

在查看了`$testing`是否为空之后，使用`=`符号将值赋给了`$testing`，然后，使用相应的`is_*`函数来测试该变量。在第5行，一个整数赋给了`$testing`变量，这是一个整数或实数。简单地说，我们可以把一个整数看做是没有小数点的数字。在第8行，一个字符串赋给了`$testing`变量，这是一个字符的集合。当你在自己的脚本里使用字符串的时候，它们总是用单引号或者双引号括起来（`'`或`"`）。在第11行，一个双精度浮点数赋给了`$testing`变量，这是一个浮点数（即，一个包含小数点的数字）。在第14行，一个布尔值赋给了`$testing`变量，它的值是两个指定值（`true`或`false`）中的一个。在第17行，使用`array()`函数创建了一个数组，我们将在第8章详细学习数组。这个特定的数组包含了3个数据项，并且脚本正确地报告`$testing`具有“数组”类型。

从第20行到脚本的末尾，没有值重新赋给`$testing`，只有类型测试。第20行和第22行分别测试`$testing`是否是一个数字或资源类型，如果不是就不会向用户显示值。脚本在第24行再次测试`$testing`是否是一个数组，如果是就显示值1。

## 5.2.1 使用settype()来改变变量的数据类型

PHP也提供了函数settype(), 用来改变一个变量的类型。要使用这个函数, 你需要在括号中放入要修改类型的变量以及要改变的目标类型, 并且用逗号隔开这两个元素, 如下所示。

```
settype($variabletochange, 'new type');
```

程序清单5.2把值3.14（一个浮点数）转换为我们在本章中所提到的4种其他标准类型。

程序清单5.2 使用settype()修改一个变量的类型

```
1: <?php
2: $undecided = 3.14;
3: echo "is ".$undecided." a double? ".is_double($undecided)."<br/>"; // double
4: settype($undecided, 'string');
5: echo "is ".$undecided." a string? ".is_string($undecided)."<br/>"; // string
6: settype($undecided, 'integer');
7: echo "is ".$undecided." an integer? ".is_integer($undecided)."<br/>"; //
   integer
8: settype($undecided, 'double');
9: echo "is ".$undecided." a double? ".is_double($undecided)."<br/>"; // double
10: settype($undecided, 'bool');
11: echo "is ".$undecided." a boolean? ".is_bool($undecided)."<br/>"; // boolean
12: ?>
```

### 提示:

根据PHP手册, 在浮点数的情况下返回“double”, 而不是直接返回“float”。我们所看到的情况也确实如此。

在每个实例中, 我们使用了相应的is\_\*函数来确保新的数据类型, 并且还使用echo把变量\$undecided的值输出到浏览器。当我们在第6行把字符串“3.14”转换为一个整数时, 小数点后面的所有信息都永久地丢失了。这就是为什么当我们在第8行将其改回double类型的时候,

\$undecided中包含的值是3。最后，在第10行，我们把\$undecided转换为一个布尔值，任何非0的数字转换为布尔值时都会变为true。当我们在PHP中显示一个布尔值的时候，true显示为1而false显示为一个空字符串，因此，在第11行，\$undecided显示为1。

把上述代码放入到一个名为 settype.php 的文本文件中，并且把这个文件放到你的 Web服务器的文档根目录下，当我们通过Web浏览器访问这个脚本的时候，会产生如下的输出。

```
is 3.14 a double? 1
is 3.14 a string? 1
is 3 an integer? 1
is 3 a double? 1
is 1 a boolean? 1
```

提示：

你将不会看到像前面小节中所见到的那样一条关于未定义变量的注意，因为在这段脚本的开始，定义了\$undecidedis变量并给它赋了一个值。

## 5.2.2 通过类型转换改变变量的数据类型

使用settype()改变一个已有变量的类型和使用类型转换改变变量类型的主要区别在于，类型转换会生成一个拷贝，而保持原来的变量不动。要通过类型转换来改变类型，我们首先在括号中，在所要复制的变量的前面，给出一种数据类型的名字。例如，如下的一行代码创建了\$originalvar变量的一个副本，它具有一个指定的类型（整数）和一个新的名字\$newvar。\$originalvar变量仍然可用，并且仍保留原来的类型。\$newvar是一个全新的变量。

```
$newvar = (integer) $originalvar
```

程序清单5.3展示了通过类型转换改变数据类型的例子。

程序清单5.3 对一个变量进行类型转换

```
1: <?php
2: $undecided = 3.14;
3: $holder = (double) $undecided;
4: echo "is ".$holder." a double? ".is_double($holder)."<br/>"; // double
5: $holder = (string) $undecided;
6: echo "is ".$holder." a string? ".is_string($holder)."<br/>"; // string
7: $holder = (integer) $undecided;
8: echo "is ".$holder." an integer? ".is_integer($holder)."<br/>"; // integer
9: $holder = (double) $undecided;
10: echo "is ".$holder." a double? ".is_double($holder)."<br/>"; // double
11: $holder = (boolean) $undecided;
12: echo "is ".$holder." a boolean? ".is_bool($holder)."<br/>"; // boolean
13: echo "<hr/>";
14: echo "original variable type of $undecided: ";
15: echo gettype($undecided); // double
16: ?>
```

我们不会真的改变\$undecided变量的类型，它在这个脚本中始终保持为double类型，第15行说明了这一点，在那里，我们使用gettype()函数来确定\$undecided的类型。

**提示：**

不要使用gettype()来测试某个变量的数据类型，尽管在这里是这么做的，因为这可能很慢并且该函数可能在未来的版本中被弃用。在正式的情况下，使用is\_\*函数来测试变量的数据类型。这里使用gettype()函数是为了便于说明。

实际上，通过对\$undecided进行类型转换，我们创建了一个副本，然后将其转换为在类型转换时所指定的类型，并且存储到变量\$holder中。这个类型转换首先发生在第3行，随后还发生在第5行、第7行、第9行和第11行。由于我们只是使用\$undecided的一个副本而不是最初的变量，\$undecided不会失去其最初的值，这和在程序清单5.2中第6行

\$undecided变量的类型从字符串转换为整数的情况不一样。

把程序清单5.3的内容放入到一个名为casttype.php的文本文件，然后把这个文件放入到你的Web服务器文档根目录下。当你通过Web浏览器访问这个脚本的时候，它会产生如下的输出。

```
is 3.14 a double? 1
is 3.14 a string? 1
is 3 an integer? 1
is 3.14 a double? 1
is 1 a boolean? 1
original variable type of 3.14: double
```

现在你已经了解了如何使用settype()函数或通过类型转换把一个变量从一种类型改变为另一种类型，考虑一下为什么这种转换可能会有用。这并不是一个必须经常使用的过程，因为在脚本内容需要类型转变的情况下，PHP会自动为你进行变量类型转换。然而，自动的类型转换是临时的，并且你可能想让一个变量持久地保存一种特定的数据类型，因此，就需要明确地改变类型。

例如，一个用户从一个HTML表单中输入数字，这个数字将要供脚本作为“字符串”类型使用。如果试图把两个字符串相加，由于它们包含数字，在相加的时候，PHP会帮忙把这些字符串转换为数字。因此

```
"30cm" + "40cm"
```

将会产生一个结果70。

#### 提示：

通用术语“数字”在这里指的是整数和浮点数。如果用户输入是浮点数的形式，并且相加的字符串为“3.14cm”和“4.12cm”，答案将会是7.26。在把字符串类型转换为整数或浮点数的时候，PHP将会忽略任何非数字字符，字符串将会被截断，并且，从第一个非数字字符开始向后

的所有字符都被忽略。因此，字符串“30cm”转换为“30”，字符串“6ft2in”变成6，因为字符串剩余部分都将计算为0。

你可能想要自己清空用户输入，并且在脚本中用特殊的方式使用它。假设已经要求用户提交一个数字，我们可以通过声明一个变量并且把用户的输入赋给它来模拟这种情况。

```
$test = "30cm";
```

正如你所见到的，用户已经为他们的数字添加了单位，用户输入的是“30cm”，而不是“30”。可以通过将其类型转换为一个整数来确保用户输入是整齐的。

```
$newtest = (integer) $test;  
echo "Your imaginary box has a width of $newtest centimeters.";
```

最终的输出如下。

```
Your imaginary box has a width of 30 centimeters.
```

如果用户的输入没有进行类型转换，并且当显示有关一个盒子的宽度的语句时，最初的变量\$test的值将替代\$newtest，结果如下。

```
Your imaginary box has a width of 30cm centimeters.
```

这个输出显得奇怪，实际上，它看上去只是重复了未经整理过的（原始的）用户输入。

### 5.2.3 为何测试类型

为什么知道变量的类型可能会有用呢？在编程中经常会有这样的情况，传递给你的数据是来自于另外一个来源。在第7章中，你将会学习如何在自己的脚本中创建函数，以及数据常常在一个或多个函数之间传

递，因为它们可以以参数的形式从调用代码接受信息。对于要使用给定数据类型的函数，首先验证函数被给定的值具有正确的数据类型是一个不错的主意。例如，期待拥有一个“资源”类型的数据的一个函数，当传递给它一个字符串的时候，它是无法正常工作的。

## 5.3 操作符和表达式

根据我们目前学习的知识，我们可以把数据赋给变量，甚至可以了解变量的数据类型并改变它。然而，除非你可以操作已经存储的数据，否则，一种编程语言还并不是非常有用。操作符（operator）就是用来操作存储在变量中的数据的符号，从而使得以下情况成为可能：用一个或多个值来产生一个新的值，或者在一个条件下检查数据的有效性以便确定下一个步骤等等。操作符在其上操作的值叫做操作数（operand）。

### 提示：

操作符是一个符号或者一系列的符号，当它用来连接值的时候，执行一个操作并且通常会产生一个新值。操作数是用来和一个操作符连接的一个值。一个操作符通常有两个或更多个操作数。

在下面的简单例子中，两个操作数和一个操作符组合到一起产生一个新的值。

$(4 + 5)$

整数4和5是操作数。这些操作数通过一个加号操作符（+）来操作，产生整数9。操作符几乎总是位于两个操作数之间，尽管你会在本章的后面看到少数例外情况。

操作数和一个操作符组合到一起得到一个结果就叫做表达式（expression）。尽管操作符及其操作数构成了表达式的基础，但一个



表达式不一定要包含操作符。实际上，PHP中的表达式定义为可以用作一个值的任何事物。这包括像654这样的整数常数，像\$user这样的变量，以及像gettype()这样的函数调用。例如，表达式（4+5）由两个表达式“（4”和“5）”以及一个操作符“+”组成。当一个表达式产生一个值时，通常说求得该值。也就是说，当考虑到所有的子表达式的时候，表达式可以看做好像是该值本身的代码。在这个例子中，表达式（4+5）求得值9。

**提示：**

一个表达式可以是函数、值和求值的操作符的任意组合。首要的原则是，如果你可以将其作为一个值使用，它就是一个表达式。

既然我们已经解决了原理性问题，现在可以看看操作符在PHP程序设计中的一般用法。

### 5.3.1 赋值操作符

在以前的例子中声明一个变量的时候，我们已经见到过赋值操作符的几次使用，赋值操作符由单个字符组成，即=。赋值操作符取右操作数的值并将其赋给左操作数，示例如下。

```
$name = "Jimbo";
```

\$name变量现在包含了字符串“jimbo”。这个结构也是一个表达式，只不过第一眼看上去好像赋值操作符只是改变了\$name变量的值，而没有产生一个新的值。实际上，使用赋值操作符的语句总是求得右操作数的值的一个副本。因此，下述代码

```
echo $name = "Jimbo";
```

向浏览器显示字符串“jimbo”，同时还把值“jimbo”赋给\$name变量。

### 5.3.2 算术操作符

算术操作符做我们所期待的事情，即执行算术运算。表5-2列出了这些操作符，并举例说明它们的用法和结果。

表5-2 算术操作符

操作符名称	例 子	示 例 结 果
+加法	10+3	13
-减法	10-3	7
/除法	10/3	3.33333333333333
*乘法	10*3	30
%模除	10%3	1

加法操作符把右操作数增加到左操作数上；减法操作符把右操作数从左操作数上减去；除法操作符用左操作数除以右操作数；乘法操作符用左操作数乘以右操作数；模除操作符返回左操作数除以右操作数的余数。

### 5.3.3 连接操作符

连接操作符用一个句点（.）表示。它把两个操作数都当作是字符串，把右操作数附加到左操作数上。如

```
"hello"." world"
```

返回

```
"hello world"
```

注意，单词之间最终有一个空格，这是因为在第二个操作数（是“world”而不是“world”）的前面有一个空格。连接操作符逐字把两个字符串连接起来而不添加任何空白。因此，如果你想要连接两个头部和尾部都没有空格的字符串的话，例如

```
"hello"."world"
```

会得到如下的结果。

```
"helloworld"
```

不管和连接操作符一起使用的操作数是什么数据类型，它们都会被当作字符串对待，并且结果也总是字符串类型。在本书中我们会频繁地用到连接操作符，当我们需要把某种类型的表达式结果和一个字符串组合到一起的时候使用，示例如下。

```
$cm = 212;  
echo "the width is ".$cm/100)." meters";
```

### 5.3.4 复合赋值操作符

虽然只有一个真正的赋值操作符，但是PHP提供了一些复合赋值操作符来改变左操作数并返回一个结果，同时也修改了变量最初的值。按照规则，操作符使用操作数但不会改变其最初的值，但是复合赋值操作

符打破了这一规则。复合赋值操作符由一个标准操作符后面跟着一个等号构成。复合赋值操作符为你省去了在脚本中分两步来使用两个操作符的麻烦。例如，如果有一个值为4的变量，并且想要再把这个值加4，你可能会看到如下实现。

```
$x = 4;  
$x = $x + 4; // $x now equals 8
```

然而，我们也可以使用一个复合赋值操作符（+=）来加和并返回新值，如下所示。

```
$x = 4;  
$x += 4; // $x now equals 8
```

每个算术操作符包括连接操作符都有相应的复合赋值操作符。表5-3列出了这些复合赋值操作符并给出了其用法示例。

表5-3 一些复合赋值操作符

操 作 符	示 例	等 同 于
+=	\$x+=5	\$x=\$x+5
--	\$x-=5	\$x=\$x-5
/=	\$x/=5	\$x=\$x/5
*=	\$x*=5	\$x=\$x*5
%=	\$x%=5	\$x=\$x%5

.=	\$x.=“test”	\$x=\$x.“test”
----	-------------	----------------

表5-3中的每个例子使用右操作数的值来改变\$x的值。此后再用到\$x的时候将引用新值。示例如下。

```
$x = 4;
$x += 4; // $x now equals 8
$x += 4; // $x now equals 12
$x -= 3; // $x now equals 9
```

这些操作符将会在本书的很多脚本中用到。当你想创建一个动态文本的时候，将会频繁地看到复合赋值操作符。遍历一个脚本并为一个字符串添加内容，例如，动态地构建HTML代码来显示一个表，是复合赋值操作符用法的主要例子。

### 5.3.5 自动增加和减少一个整型变量

用PHP编码的时候，你将经常会发现有必要对一个整型变量加1或减1。当你计算一个循环的次数的時候，通常需要这么做。你已经学习了这么做的两种方式，即使用加法操作符来增加\$x的值：

```
$x = $x + 1; // $x is incremented by 1
```

或者使用一个复合赋值操作符。

```
$x += 1; // $x is incremented by 1
```

在这两个例子中，新的值都赋给了\$x。因为这种自动增加或减少的表达式很常见，PHP提供了某些专门的操作符以允许你对一个整型变量增加或减少整数常量1，再把结果赋给变量自身。这就是所谓的后自增（post-increment）和后自减（post-decrement）。后自增操作符包含跟在变量名后面的两个加号，如下所示。

```
$x++; // $x is incremented by 1
```

这个表达式把变量\$x所表示的值增加了1。以同样的方式使用两个减号符号将会减小该变量的值。

```
$x--; // $x is decremented by 1
```

如果和一个条件操作符一起使用后自增或后自减操作符，那么，只有在第一个操作完成以后操作数才会被修改。

```
$x = 3;  
$y = $x++ + 3;
```

在这种情况下，\$y首先变为6(3 + 3)，然后\$x增加1。

在某些情况下，在一个测试表达式中，你可能希望在测试执行之前把一个变量增加或减少1。PHP为此提供了前自增（pre-increment）和前自减（pre-decrement）操作符。这两个操作符按照与后自增和后自减操作符相同的方式工作，但是，它们把加号或减号放在了变量的前面。

```
++$x; // $x is incremented by 1  
--$x; // $x is decremented by 1
```

如果这些操作符用作一个测试表达式的一部分，变量加1将会在测试执行之前执行。例如，在下面的代码段中，在测试\$x是否小于4之前，它先增加1。

```
$x = 3;  
++$x < 4; // false
```

测试表达式返回false，因为\$x首先加1变为4，而4不再小于4了。

### 5.3.6 比较操作符

比较操作符对使用它们的操作数进行比较测试，并且如果测试成功返回布尔值true，如果测试失败返回布尔值false。当在脚本中使用控制结构的时候，例如if和while这样的语句，这种类型的表达式很有用。我们将会在第6章介绍if和while语句。

例如，要测试\$*x*中包含的值是否小于5，我们可以使用小于操作符作为表达式的一部分，示例如下。

```
$x < 5
```

如果\$*x*包含的值是3，这个表达式的值将为true。如果\$*x*包含的值是7，这个表达式将会得到false。

表5-4列出了比较操作符。

表5-4      比较操作符

操 作 符	名 称	如果...则返回true	示例(\$ <i>x</i> 等于4)	结 果
==	等于	左边等于右边	<i>\$x</i> ==5	false
!=	不等于	左边不等于右边	<i>\$x</i> !=5	true
===	等同	左边等于右边并且它们类型相同	<i>\$x</i> ===4	true
	不等同	左边等于右边但是它们类型不同	<i>\$x</i> ===“4”	false
>	大于	左边大于右边	<i>\$x</i> >4	false

>=	大于或等于	左边大于或等于右边	\$x>=4	true
<	小于	左边小于右边	\$x<4	false
<=	小于或等于	左边小于或等于右边	\$x<=4	true

这些操作符经常和整数或双精度数一起使用，尽管等于操作符也用来比较字符串。一定要理解==操作符和=操作符之间的不同。== 操作符测试相等性，而=操作符赋值。另外，别忘了，===测试值以及类型的相等性。

### 5.3.7 使用逻辑操作符创建复杂的测试表达式

逻辑操作符测试布尔值的组合。例如，or操作符用两条竖线(||)或者一个单词or来表示，如果左操作数或右操作数中有一个为true，结果返回布尔值true。

```
true || false
```

这个表达式返回true。

and操作符用两个&符号(&&)表示或者直接使用单词and来表示，如果两个操作数都是true，它就返回布尔值true。

```
true && false
```

这个表达式返回布尔值false。你不太可能只使用逻辑操作符来测试布尔常量，因为测试两个或多个表达式来得到一个布尔值更有意义，示



例如下。

```
($x > 2) && ($x < 15)
```

如果\$*x*包含了一个大于2而小于15的值，返回布尔值true。当比较表达式的时候可以使用括号，以便代码更容易阅读并能够表示表达式求值的优先级。表5-5列出了逻辑操作符。

表5-5     逻辑操作符

操 作 符	名 称	如果...则返回true	示 例	结 果
	Or	左边或右边为true	true    false	true
or	Or	左边或右边为true	true or false	true
xor	Xor	左边或右边为true，但不都为true	true xor true	false
&&	And	左边和右边都为true	true && false	false
and	And	左边和右边都为true	true and false	false
!	No	操作数不为true	! true	false

你可能会奇怪为什么or和and操作符都有两个版本，这是个不错的问题。答案是取决于操作的优先级，我们将会在后面来介绍这一点。

5.3.8    操作符优先级

当我们在一个表达式中使用一个操作符的时候，PHP引擎通常从左到右读取表达式。对于使用多个操作符的复杂表达式，如果没有任何说明，PHP引擎将会被引入歧途。例如，首先考虑如下一个简单的例子。

`4 + 5`

这里不会引起混淆，PHP只是把4和5相加。但是，对于下面的具有两个操作符的代码。

`4 + 5 * 2`

这就产生了一个问题。PHP是应该先对4和5求和，然后把结果乘以2，得到18呢？还是应该把4和5乘以2的结果相加，得到14呢？如果我们只是简单地从左到右地读取，前一个结果是正确的。然而，PHP对不同的操作符附加了不同的优先级，并且由于乘法操作符比加法操作符具有更高的优先级，这个问题的第二个解答是正确的，即把4和5乘以2的结果相加。

然而，你可以在表达式的外围放置一对括号，从而覆盖掉操作符本身的优先级。在下面的代码段中，加法表达式会在乘法表达式之前计算。

`(4 + 5) * 2`

不管一个复杂表达式中的操作符的优先级是怎样的，使用括号来让代码更为清楚并且避免掉一些错误（例如在购物车应用中对错误的小计算销售税），这是个不错的想法。下面是本章所介绍的操作符按照优先级高低的一个列表（具有较高优先级的操作符列在前面）。

```
++, --, (cast)
/, *, %
+, -
<, <=, ==, >
==, ===, !=
&&
||
=, +=, -=, /=, *=, %=, .=
and
xor
or
```

正如你所见到的，or比||的优先级低，并且and比&&的优先级低，因此，可以使用较低优先级的逻辑操作符来改变一个复杂测试表达式的读取方式。在下面的代码段中，两个表达式是等同的，但是后者更容易理解。

```
$x and $y || $z

$x && ($y || $z)
```

更进一步地考虑，如下的代码段更容易理解。

```
$x and ($y or $z)
```

然而，这三个例子都是等同的。

优先级的顺序是PHP中&&和and都存在的唯一的原因。对于||和or，也是如此。在大多数情况下，和那些依靠这些操作符的优先级顺序的代码比较，使用括号会让代码更清楚并且更少出错。在本书中，我们将倾向于使用更为常见的||和&&操作符，并且依靠括号来设置特定的

操作符顺序。

## 5.4 常量

变量为存储数据提供了一种灵活的方式，因为我们可以执行脚本的过程中随时改变它们所存储的值及其类型。然而，如果你想要使用一个在整个脚本执行过程中必须保持不变的值的话，可以定义并使用一个常量（`constant`）。必须使用PHP内建的`define()`函数来创建一个常量，随后这个常量是不能改变的，除非再次明确地`define()`它。要使用`define()`函数，把常量的名字以及你希望赋给它的值放入到括号中，中间用逗号隔开，示例如下。

```
define("YOUR_CONSTANT_NAME", 42);
```

你想要设置的这个值可以是一个数字、一个字符串或者一个布尔值。按照惯例，常量的名字应该大写的，常量只能使用常量名访问，不需要美元符号。程序清单5.4展示了如何定义和访问一个常量。

程序清单5.4 定义和访问一个常量

```
1: <?php
2: define("THE_YEAR", "2012");
3: echo "It is the year ".THE_YEAR;
4: ?>
```

### 提示：

使用常量的时候，别忘了它们可以用在脚本中的任何地方，例如包含在脚本内的一个外围函数中。

注意，我们在第3行使用了连接操作符把常量所存储的值附加到字

字符串“It is the year”的后面，因为PHP不会区分一个常量和引号之间的一个字符串。

把上述代码放入到一个名为constant.php的文本文件中，然后把该文件放置到Web服务器的文档根目录下。当你通过Web浏览器访问这个脚本的时候，会得到如下的输出。

```
It is the year 2012
```

define()函数也可以接受第三个布尔参数来确定常量名是否应该区分大小写。默认情况下，常量名是区分大小写的。然而，通过把true参数传递给define()函数，我们可以改变这一行为，因此，如果我们可以建立新的THE\_YEAR常量，示例如下。

```
define("THE_YEAR", "2012", true);
```

我们可以访问它的值而不用担心大小写，示例如下。

```
echo the_year;  
echo ThE_YeAr;  
echo THE_YEAR;
```

上述的3个表达式是等同的，并且它们都将得到同一个输出，即2012。对于那些使用我们的代码的其他程序员，这个功能可以使得脚本更为友好一些，因为他们访问我们已经定义的一个常量的时候，不需要考虑大小写。但是另一方面，通常其他的常量是区分大小写的，这可能增加了而不是减少了程序员的混淆，因为他们可能忘记应该以何种方式来对待哪个常量。除非你有足够充分的理由这么做，否则的话，还是保持常量区分大小写并且使用大写字母来定义它们才是最安全的，这是一个便于记住的约定，更别说这是标准方式了。

预定义常量是PHP自动为你提供的一些内建常量。例如，常量 `__FILE__` 返回PHP引擎当前所读取的文件的名字。常量 `__LINE__` 返回该文件当前的行数。这是所谓的“神奇常量”的两个例子，因为他们不是静态地预定义的，并且会根据使用它们的环境而变化。要获得预定义常量的完整列表，参见

<http://www.php.net/manual/en/language.constants.predefined.php> 。

你也可以通过 `PHP_VERSION` 常量来了解PHP的哪个版本在解释脚本。发布一个错误报告时，如果需要把版本信息包含在脚本输出中，这个常量会很有用。`PHP_VERSION` 常量是一个预定义常量（并且是一个保留字）。要获取保留字常量的完整列表，参见

<http://www.php.net/manual/en/reserved.constants.php> 。

## 5.5 小结

在本章中，我们介绍了PHP语言的一些基本功能。了解了变量以及如何使用赋值操作符来为它们赋值，介绍了变量和内建的超全局变量的作用域。本章还介绍了操作符，并且介绍了如何把一些最常见的操作符组合到表达式中。最后，我们学习了如何定义和访问常量。

既然已经掌握了PHP的一些基础，下一章将带你真正体验PHP编程。你将学习如何编写脚本，以及在变量、表达式和操作符的帮助下，做出流程判断和重复执行任务。



## 5.6 Q&A

**Q:** 为什么知道一个变量的数据类型是有用的？

**A:** 一个变量的数据类型往往限制了你可以对它做什么。例如，你不能对一个简单的字符串执行和数组相关的功能。类似的，在算术计算中使用一个变量之前，可能想要确保它包含一个整数或者浮点数的值，即便在此情况下PHP常常改变数据类型以帮助你。

**Q:** 命名变量的时候应该遵守什么惯例？

**A:** 目标应该总是确保代码易于阅读和理解。像\$ab123245这样的一个变量，对于你了解它在脚本中的作用没有任何帮助，而且容易导致输入错误。保持变量名简单而且具有描述性。当你大概一个月后再回头来看代码的时候，变量名\$f可能对你没多大意义。而变量名\$filename应该更有意义。

**Q:** 我应该了解操作符优先级表吗？

**A:** 没什么理由说不应该去了解，但是如果有更重要的任务，我情愿偷懒过去。在定义自己的优先级顺序的时候，通过在表达式中使用括号，你可以使自己的代码更容易阅读。

## 5.7 实践练习

这个实践练习设计用来帮助你预测可能遇到的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 下列变量中的哪一个不是合法的？

```
$a_value_submitted_by_a_user  
$666666xyz  
$xyz666666  
$_____counter_____  
$the first  
$file-name
```

2. 下面这段代码的输出是什么？

```
$num = 33;  
(boolean) $num;  
echo $num;
```

3. 如下语句的输出是什么？

```
echo gettype("4");
```

4. 如下代码段的输出是什么？

```
$test_val = 5.5466;  
settype($test_val, "integer");  
echo $test_val;
```

5. 如下的哪条语句没有包含一个表达式？

```
4;  
is_int(44);  
5/12;
```

6. 问题5中的哪条语句包含了一个操作符？

7. 下面的表达式将返回什么值？

```
5 < 2
```

返回值的数据类型是什么？

## 解答

1. 变量名\$666666xyz不合法，因为它没有以字母或下划线开头。变量名\$the first不合法，因为它包含一个空格。\$file-name也不合法，因为它包含一个非字母的字符（-）。
2. 这段代码将会显示整数33。类型转换会产生存储在\$num中的值的一个转换后的副本。它不会修改实际存储在变量里的初始值。
3. 这条语句将会输出字符串“string”。
4. 这段代码将输出值5。当一个浮点数转换为一个整数的时候，小数点后的任何信息都会丢失。
5. 它们都是表达式，因为它们都会求得值。
6. 语句5/12包含一个除法运算符。
7. 这个表达式将会得到false，这是一个布尔值。

## 思考题

1. 创建一个脚本，至少包含5个不同的变量。用不同数据类型的值填充它们，并且使用`gettype()`函数在浏览器中显示出每种类型。

2. 给两个变量赋值。使用比较操作符来测试第一个值是否和第二个值相同。

- 与第二个值相同。
- 小于第二个值。
- 大于第二个值。
- 小于或等于第二个值。

在浏览器中显示出每次测试的结果。

修改赋给测试变量的值，并且再次运行脚本。

## 第6章 PHP的流程控制功能

在本章中你将学习：

- 如果一个测试表达式结果为**true**，如何使用**if**语句执行代码。
- 当一条**if**语句的测试表达式计算为**false**时，如何执行供选择的代码块。
- 如何使用**switch**语句根据测试表达式返回的值来执行代码。
- 如何使用**while**语句来重复执行代码。
- 如何使用**for**语句使循环更简洁。
- 如何跳出循环。
- 如何将一个循环嵌套到另一个循环中。
- 如何在控制结构中使用**PHP**的**start**和**end**标签。

前面的章节中创建的脚本只是沿着一个方向顺序执行。也就是说，每次脚本运行的时候，同样的语句按照同样的顺序在执行。这样就没有多少灵活性，因为各种动态程序设计都至少需要一个或两个循环，更不必说在顺序执行之前检查某些条件的正确性的能力。现在我们将学习程序设计结构，这些结构使得脚本适合于实际环境。

## 6.1 转换流程

计算条件并因此改变自己的操作，这对于脚本来说是很常见的。这些判断使PHP页面变得动态——也就是说，能够根据情况改变输出。和大多数程序设计语言一样，PHP允许你用if语句来做到这一点。

### 6.1.1 if语句

if语句是一种控制跟在其后面的语句（即一个单个的语句或在括号内的代码块）执行的方法。if语句计算括号之内的表达式——如果这个表达式结果为true，执行后续语句，否则完全跳过语句。这个功能允许脚本根据多种因素来做判断。

```
if (expression) {  
    // code to execute if the expression evaluates to true  
}
```

只有当变量包含字符串“happy”时，程序清单6.1才执行代码块。

程序清单6.1 if语句

---

```
1: <?php  
2: $mood = "happy";  
3: if ($mood == "happy") {  
4:     echo "Hooray! I'm in a good mood!";  
5: }  
6: ?>
```

---

在第2行，把“happy”赋给变量\$mood。在第3行，比较运算符==把变量\$mood的值与字符串“happy”做比较。如果它们匹配，表达式计算为true，接着执行后续代码，直到遇到结束的花括号（在本例中的第5行）。



把这些代码放到名为testif.php的文本文件中，并把文件放到Web服务器文档根目录下。当通过Web浏览器访问这个脚本时，产生如下的输出。

```
Hooray! I'm in a good mood!
```

如果你把\$mood的值改为“sad”或者除了“happy”以外的其他字符串，再次运行这个脚本，if语句中的表达式将计算为false，导致忽略后续代码块。这个脚本什么都不做，它将引导我们到else子句。

### 6.1.2 使用else子句的if语句

当使用if语句时，你可能想定义一个可供选择的代码块，如果测试表达式计算为false，那么执行这个代码块。通过把else添加到后面跟着其他代码块的if语句中，可以实现这一点。

```
if (expression) {  
    // code to execute if the expression evaluates to true  
} else {  
    // code to execute in all other cases  
}
```

程序清单6.2完善了程序清单6.1中的示例，因此，如果\$mood的值不等于“happy”将执行一个默认代码块。

程序清单6.2 使用else的if语句

---

```
1: <?php  
2: $mood = "sad";  
3: if ($mood == "happy") {  
4:     echo "Hooray! I'm in a good mood!";  
5: } else {  
6:     echo "I'm in a $mood mood.";  
7: }  
8: ?>
```

---

把这些代码放到名为testifelse.php的文本文件中，并将文件放到Web服务器文档根目录下。当使用Web浏览器访问这些脚本时，产生如下的输出。

```
I'm in a sad mood.
```

注意，在第2行，\$mood被赋予的值是字符串“sad”，显然不等于“happy”，所以第3行的if语句的表达式计算为false。这导致跳过了第一个代码块（第4行），而执行else后面的代码块并显示另一个供选择的信息：I’m in a sad mood。字符串“sad”是赋给变量\$mood的值。

使用一个与if语句联合的else子句，允许脚本进行有关代码执行的判断。然而，这种选择受限于either-or分支：要么执行跟在if语句后面的代码块，要么执行跟在else语句后面的代码块。现在我们将学习用于计算多个表达式的另一种选择，这些表达式是一个接一个的。

### 6.1.3 使用带有elseif子句的if语句

提供一个默认代码块（elseif部分）之前，你可以使用一个if...elseif...else子句来测试多个表达式（if...else部分）。

```
if (expression) {  
    // code to execute if the expression evaluates to true  
} elseif (another expression) {  
    // code to execute if the previous expression failed  
    // and this one evaluates to true  
} else {  
    // code to execute in all other cases  
}
```

如果最初的if表达式结果不为true，那么将跳过第一个代码块。elseif子句给出另一个表达式来计算，如果这个表达式计算为true，则执行它对应的代码块。否则，执行与else子句关联的代码块。你可以包含

任意多个elseif子句，并且如果不需要一个默认操作，可以省略else子句。

**提示：**

elseif子句也可以写成两个单词（else if）。采用哪种语法是个人喜好问题，但是PEAR（PHP extension and application repository，PHP扩展与应用库）和PECL（PHP extension community library，PHP扩展公用库）所采用的编码标准都使用elseif。

程序清单6.3在前一个示例中添加了elseif子句。

程序清单6.3 使用else和elseif的if语句

```
1: <?php
2: $mood = "sad";
3: if ($mood == "happy") {
4:     echo "Hooray! I'm in a good mood!";
5: } elseif ($mood == "sad") {
6:     echo "Awww. Don't be down!";
7: } else {
8:     echo "I'm neither happy nor sad, but $mood.";
9: }
10: ?>
```

把\$mood变量再次赋值为“sad”，如第2行所示。因为这个值与“happy”不相等，所以第4行的代码被跳过。第5行的elseif子句检测\$mood的值与“sad”是否相等，在这个示例中结果为true，因此执行第6行的代码。从第7行到第9行，提供了一个默认操作，如果前面的测试条件全都是false，那么将调用这个操作。在这个示例中，我们只简单地显示一条消息，其中包含\$mood变量的实际值。

把上述代码放到名为testifelseif.php的文本文件中，并把文件放到Web服务器文档根目录下。当通过Web浏览器访问这个脚本时，产生如

下的输出。

```
Awww. Don't be down!
```

将\$mood的值改为“iffy”并运行这个脚本，将产生如下的输入。

```
I'm neither happy nor sad, but iffy.
```

### 6.1.4 switch语句

switch语句是另一种根据表达式的结果改变流程的方法。正如你已经看到的，使用if和elseif语句可以计算多个表达式。然而，switch语句只计算一个表达式列表，基于匹配代码的一个特定位来选择正确的一个。作为if语句一部分的表达式的结果要么是true要么是false，而switch语句的表达式部分则是连续检测很多值，希望找到一个匹配的值。

```
switch (expression) {  
    case result1:  
        // execute this if expression results in result1  
        break;  
    case result2:  
        // execute this if expression results in result2  
        break;  
    default:  
        // execute this if no break statement  
        // has been encountered hitherto  
}
```

switch语句中的表达式通常只是一个变量，例如\$mood。在switch语句中，你会发现很多的条件语句。每一个条件都检测一个值是否与switch表达式的值匹配。如果条件值等于表达式的值，执行条件语句中的代码。最后，break语句将结束switch语句的执行。

如果省略了break语句，那么将顺序执行下一个条件语句，而不管前一个匹配的值是否已经找到。如果在执行到可选的默认语句之前还没

有找到一个匹配的值，那么执行默认语句。

**注意：**

在作为条件语句的一部分执行的任何代码的最后，包含一条break语句，这是很重要的。没有break语句，程序流程将继续下一个条件语句，并最终到达默认语句。在大多数情况下，这将导致一个无法预计的结果，而这个结果很可能是错误的！

程序清单6.4使用switch语句重新创建了if语句示例的功能。

程序清单6.4 switch语句

```
1: <?php
2: $mood = "sad";
3: switch ($mood) {
4:     case "happy":
5:         echo "Hooray! I'm in a good mood!";
6:         break;
7:     case "sad":
8:         echo "Awww. Don't be down!";
9:         break;
10:    default:
11:        echo "I'm neither happy nor sad, but $mood.";
12:        break;
13: }
14: ?>
```

在第2行把\$mood变量的值初始化为“sad”。第3行的switch语句将这个变量作为它的表达式。第4行的第一个条件语句检测“happy”与变量\$mood的值是否相等。在本示例中两者并不匹配，所以脚本执行到第7行的第二个条件语句。字符串“sad”与\$mood变量的值相等，所以执行这个代码块。第9行的break语句结束这个过程。第10行到第12行提供了一个默认操作，当前面的条件结果没有一个为true时执行它。

把上述代码放到名为testswitch.php的文本文件中，并把文件放到

Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如下的输出。

```
Awww. Don't be down!
```

把\$mood的值改为“happy”并运行脚本，将产生如下的输出。

```
Hooray! I'm in a good mood!
```

为了强调break语句的重要性，尝试运行没有第二个break语句的这个脚本。确保把\$mood的值改回为“sad”并且运行该脚本，输出如下。

```
Awww. Don't be down! I'm neither happy nor sad, but sad.
```

这肯定不是我们想要的输出，所以要确定在适当的地方包含break语句！

### 6.1.5 使用？运算符

？或三元操作符与if语句类似，只不过它返回一个值，这个值源自使用冒号分隔开的两个表达式中的一个。这个结构为你提供了作为一个整体的三部分，因此得名三元操作符。表达式通常根据测试表达式的结果来产生返回值。

```
(expression) ? returned_if_expression_is_true : returned_if_expression_is_false;
```

如果测试表达式计算为true，返回第二个表达式的值，否则返回第三个表达式的值。程序清单6.5使用三元操作符根据\$mood的值设置变量的值。

程序清单6.5 使用？运算符

---

```
1: <?php
2: $mood = "sad";
3: $text = ($mood == "happy") ? "I am in a good mood!" : "I am in a $mood
   mood.";
4: echo "$text";
5: ?>
```

---

在第2行，把\$mood设为“sad”。在第3行，测试\$mood与字符串“happy”是否相等。因为测试返回false，所以返回三元操作符的第三个表达式的结果。

把上述代码放到名为testten.php的文本文件中，并把文件放到Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如下输出。

```
I am in a sad mood.
```

三元操作符很难理解，但是，如果你只处理两个选择并想用简洁的代码来完成，它就派上用场了。

## 6.2 循环

迄今为止，我们已经看到了脚本可以做出和执行什么代码相关的判断。脚本还可以判定执行一个代码块多少次。循环语句专门设计来允许你完成重复任务，因为它们继续操作直到达到一个指定条件或直到显式地选择退出循环。

### 6.2.1 while语句

`while`语句看上去和一个基本的`if`语句结构类似，但它有循环的能力。

```
while (expression) {  
    // do something  
}
```

和`if`语句不同，`while`语句只要在表达式结果为`true`的情况下就执行，即如果需要会一再地执行。循环中代码块的每次执行都叫做一次迭代（iteration）。在代码块中，通常会改变某个影响`while`语句的表达式值；否则，循环将永远继续。例如，可以用一个变量来计算迭代的次数，并据此采取相应的操作。程序清单6.6创建一个`while`循环，计算并显示2的倍数直到24。

程序清单6.6 while语句

---

```
1: <?php  
2: $counter = 1;  
3: while ($counter <= 12) {  
4:     echo $counter." times 2 is " . ($counter * 2) . "<br />";  
5:     $counter++;  
6: }  
7: ?>
```

---



在这个示例中，第2行初始化变量\$counter，将其值设置为1。第3行的while语句检验\$counter变量，在\$counter的值小于或等于12时，循环将继续运行。在while语句的代码块中，\$counter的值乘以2，并且把结果显示到浏览器中。在第5行，\$counter的值每次增加1。这一步非常重要，因为如果不增加变量\$counter的值，while表达式将决不会转变为false，循环将永不停止。

把上述代码放到名为testwhile.php的文本文件中，并把文件放到Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如下的输出。

```
1 times 2 is 2
2 times 2 is 4
3 times 2 is 6
4 times 2 is 8
5 times 2 is 10
6 times 2 is 12
7 times 2 is 14

8 times 2 is 16
9 times 2 is 18
10 times 2 is 20
11 times 2 is 22
12 times 2 is 24
```

### 6.2.2 do...while语句

do...while语句看起来有点像while语句在开始的地方打开了开关。两者之间的主要差别是do...while语句在表达式真值检测之前执行代码块，而不是在表达式真值检测之后执行代码块。

```
do {
    // code to be executed
} while (expression);
```

提示：

do...while语句的测试表达式应该始终以一个分号结束。

当我们希望代码块至少执行一次时，即便是while表达式计算为false的时候也如此，do...while语句就很有用。程序清单6.7创建了一个do...while语句。这个代码块至少执行一次。

程序清单6.7 do...while语句

---

```
1: <?php
2: $num = 1;
3: do {
4:     echo "The number is: ".$num."<br />";
5:     $num++;
6: } while (($num > 200) && ($num < 400));
7: ?>
```

---

do...while语句检测\$num变量是否包含一个大于200且小于400的值。在第2行，我们把\$num初始化为1，所以这个表达式的结果为false。但是，在计算表达式之前至少执行代码块一次，所以do...while语句在浏览器中显示一个单独的行。

把上述代码放到名为testdowhile.php的文本文件中，并把文件放到Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如下的输出。

The number is: 1

如果在第2行把\$num的值改为像300这样的值并运行脚本，循环显示如下。

The number is: 300

并且将随着数字递增继续显示类似的行，直到显示如下。

```
The number is: 399
```

### 6.2.3 for语句

想用for语句做的任何事情也可以用while语句来实现，但是for语句通常是达到同样效果的更高效的方法。在程序清单6.6中，我们看到了如何在while语句之外初始化变量，接着检测while语句的表达式并在后续代码块中递增变量。for语句也可以做这样的一系列事情，而且就在一个单个的代码行中。这使代码更简洁，并且较少发生类似由于忘记递增变量计数器而导致一个无限循环的情况。

```
for (initialization expression; test expression; modification expression) {  
    // code to be executed  
}
```

#### 提示：

无限循环，顾名思义，是没有限制地运行的循环。如果循环正在无限运行，脚本将运行无限长时间。这种行为对Web服务器造成了很大的压力并且导致网页不可用。

for语句的圆括号内的表达式用分号分隔开。通常，第一个表达式初始化一个计数器变量，第二个表达式是循环的检测条件，第三个表达式递增计数器变量。程序清单6.8展示了重新创建程序清单6.6中的示例的for语句，这个示例是用2乘以12个数字。

程序清单6.8 使用for语句

---

```
1: <?php
2: for ($counter=1; $counter<=12; $counter++) {
3:     echo $counter." times 2 is " . ($counter * 2) . "<br />";
4: }
5: ?>
```

---

把上述代码放到名为testfor.php的文本文件中，并把文件放到Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如下的输出。

```
1 times 2 is 2
2 times 2 is 4
3 times 2 is 6
4 times 2 is 8
5 times 2 is 10
6 times 2 is 12
7 times 2 is 14
8 times 2 is 16
9 times 2 is 18
10 times 2 is 20
11 times 2 is 22
12 times 2 is 24
```

程序清单6.6和6.8的结果完全一样，但是for语句使得程序清单6.8中的代码更加简洁。因为在语句一开始就初始化\$counter变量并递增，循环的逻辑一目了然。也就是，正如第2行看到的，第一个表达式初始化\$counter变量并赋值为1，测试表达式验证\$counter包含一个小于或等于12的值，并且最后的表达式使\$counter变量递增。这些表达式中的每一个都包含在代码的单独一行中。

当脚本执行的顺序到达for循环时，初始化\$counter变量并计算测试表达式。如果表达式的结果为true，则执行代码块。接着递增\$counter变量并再次计算测试表达式。这个过程一直持续到测试表达式的结果为false。

## 6.2.4 用break语句跳出循环

`while`语句和`for`语句都包含了一个可以用来结束循环的内建表达式。然而，`break`语句允许你根据补充测试的结果来跳出循环，这为预防错误提供了一种保护措施。程序清单6.9创建了一个简单的`for`语句，它用一个很大的数除以一个递增的变量，在屏幕上输出结果。

程序清单6.9 用递增到10的数来除4000的`for`循环

---

```
1: <?php
2: for ($counter=1; $counter <= 10; $counter++) {
3:     $temp = 4000/$counter;
4:     echo "4000 divided by ".$counter." is...".$temp."<br />";
5: }
6: ?>
```

---

在第2行，这个示例初始化`$counter`变量并赋值为1。`for`语句中的测试表达式验证`$counter`的值小于或等于10。在代码块中，4000除以`$counter`，并把结果显示到浏览器中。

把上述代码放到名为`testfor2.php`的文本文件中，并把文件放到Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如下的输出。

```
4000 divided by 1 is... 4000
4000 divided by 2 is... 2000
4000 divided by 3 is... 1333.33333333
4000 divided by 4 is... 1000
4000 divided by 5 is... 800
4000 divided by 6 is... 666.666666667
4000 divided by 7 is... 571.428571429
4000 divided by 8 is... 500
4000 divided by 9 is... 444.444444444
4000 divided by 10 is... 400
```

这似乎已经够直接了。但是如果你放入到`$counter`的值来自用户的输入呢？这个值可能是一个负数，甚至是一个字符串。让我们回头看第一个示例，用户输入了一个负数。把`$counter`的初始值从1改变为-4，当

第5次执行代码块时将发生4000除以0。对于你的程序来说，被0除通常不是一个好主意，因为这样的操作导致的结果是“undefined”。如果变量\$counter的值等于零，程序清单6.10通过跳出循环防止了这种情况的发生。

程序清单6.10 使用break语句

```
1: <?php
2: $counter = -4;
3: for (; $counter <= 10; $counter++) {
4:     if ($counter == 0) {
5:         break;
6:     } else {
7:         $temp = 4000/$counter;
8:         echo "4000 divided by ".$counter." is...".$temp."<br />";
9:     }
10: }
11 ?>
```

**提示：**

在PHP里用一个数除以零不会导致一个致命的错误。相反，PHP产生一个警告并继续执行。

如第4行所示，我们使用了一个if语句，试图在数学运算中使用这个值之前，检测\$counter的值。如果\$counter的值等于零，那么break语句立即停止执行代码块，并且程序将在for语句（第11行）之后继续执行。

把上述代码放到名为testfor3.php的文本文件中，并把文件放到Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如下的输出。

```
4000 divided by -4 is... -1000
4000 divided by -3 is... -1333.33333333
4000 divided by -2 is... -2000
4000 divided by -1 is... -4000
```

注意，变量\$counter在第2行初始化，并不在for语句中的圆括号内。这个方法通常用来模拟一种情况，其中\$counter的值在脚本之外设定。

#### 你知道吗？

你甚至可以省略for语句中的所有表达式，但是必须保留分隔的分号。

## 6.2.5 用continue语句跳过迭代

continue语句结束当前迭代的执行，但不会导致整个循环结束。相反，立即开始下一个迭代。程序清单6.10中使用break语句有一点过激，在程序清单6.11中使用continue语句，不用完全结束循环就可以避免除以0的错误。

程序清单6.11 使用continue语句

```
1: <?php
2: $counter = -4;
3: for (; $counter <= 10; $counter++) {
4:     if ($counter == 0) {
5:         continue;
6:     }
7:     $temp = 4000/$counter;
8:     echo "4000 divided by ".$counter." is...".$temp."<br />";
9: }
10: ?>
```

第5行，我们用continue语句替换了break语句。如果变量\$counter的值等于0，跳过当前迭代并立即开始下一次迭代。

把上述代码放到名为testcontinue.php的文本文件中，并把文件放到

Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如下的输出。

```
4000 divided by -4 is... -1000
4000 divided by -3 is... -1333.33333333
4000 divided by -2 is... -2000
4000 divided by -1 is... -4000
4000 divided by 1 is... 4000
4000 divided by 2 is... 2000
4000 divided by 3 is... 1333.33333333
4000 divided by 4 is... 1000
4000 divided by 5 is... 800
4000 divided by 6 is... 666.666666667
4000 divided by 7 is... 571.428571429
4000 divided by 8 is... 500
4000 divided by 9 is... 444.444444444
4000 divided by 10 is... 400
```

#### 注意：

使用break和continue语句将让代码变得更难以阅读，因为这增加了包含它们的循环语句的逻辑层次的复杂性。小心地使用这些语句，或者对展示给其他程序员（或者你自己）的代码添加注释，说清楚你使用这些语句所要达到的目的。

## 6.2.6 嵌套循环

循环还可以包含其他的循环语句，只要逻辑有效且循环完整。当动态创建HTML表格时，这些语句的组合特别有用。程序清单6.12使用两个for语句在浏览器中显示一个乘法表。

程序清单6.12 嵌套两个for循环

```
1: <?php
2: echo "<table style=\"border: 1px solid #000;\"> \n";
3: for ($y=1; $y<=12; $y++) {
4:     echo "<tr> \n";
5:     for ($x=1; $x<=12; $x++) {
6:         echo "<td style=\"border: 1px solid #000; width: 25px;
7:             text-align:center;\">";
```



```
8:         echo ($x * $y);
9:         echo "</td> \n";
10:    }
11:    echo "</tr> \n";
12: }
13: echo "</table>";
14: ?>
```

---

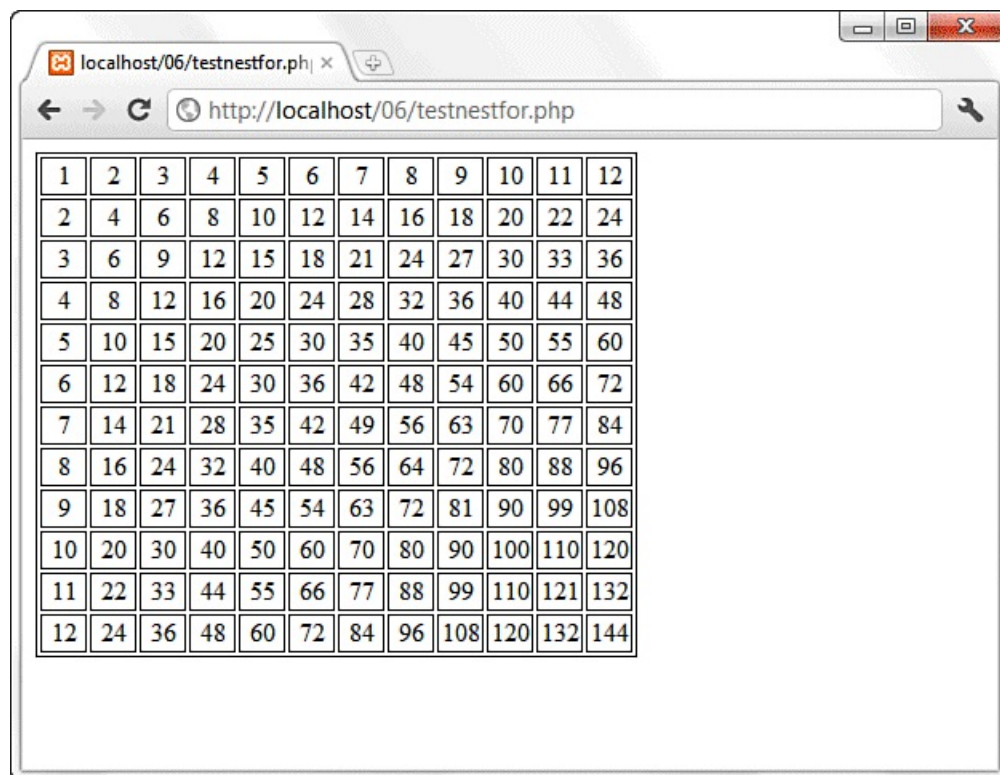
在检验for循环之前，让我们仔细看看程序清单6.12中的第2行。

```
echo "<table style=\"border: 1px solid black;\"> \n";
```

注意，在包含表格样式信息的字符串中的每个引号之前，我们使用了反斜杠（\）。这些反斜杠也出现在第6行和第7行中的表格数据单元的样式信息中。这是必要的，因为它告诉PHP引擎我们想要使用引号字符，而不是让PHP把它解释成字符串的开始或结束。如果没有用反斜杠字符将引号转义，该语句对于引擎来说就没有意义了，因为引擎将把语句理解为一个字符串后面跟着一个数字，数字后面再跟着另一个字符串。这样一个结构可能产生错误。也是在这行，我们使用\n表示换行字符，一旦将它提交给浏览器，源程序将更容易理解。

外层的for语句（第3行）初始化一个名为\$y的变量，并为它赋予初始值1。这个for语句定义一个准备验证\$y的值小于或等于12的表达式，接着定义一个将要用到的增量。在每次迭代中，代码块输出一个tr（表格行）的HTML元素（第4行）并开始内层for语句（第5行）。这个内层循环初始化一个名为\$x的变量，并且定义了和外循环一致的表达式。对于每次迭代，内循环输出一个td（表格单元格）元素到浏览器（第6行和第7行），还输出\$x乘以\$y的结果（第8行）。在第9行，我们关闭表格单元格。内层的循环完成后，我们退回到外面的循环，在第11行结束表格行，准备再次开始这个过程。当外层循环完成时，显示一个整齐的带格式的乘法表。我们在第13行通过结束表来把所有表格信息包起来。

把上述代码放到名为testnestfor.php的文本文件中，并把文件放到Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如图6-1所示的输出。



The screenshot shows a web browser window with the address bar displaying 'http://localhost/06/testnestfor.php'. The main content area displays a 12x12 grid of numbers. The numbers are arranged in a pattern where each row starts with a value and increases by 1 for each subsequent column. The values in each row are: Row 1: 1-12; Row 2: 2-13; Row 3: 3-14; Row 4: 4-15; Row 5: 5-16; Row 6: 6-17; Row 7: 7-18; Row 8: 8-19; Row 9: 9-20; Row 10: 10-21; Row 11: 11-22; Row 12: 12-23.

1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144

图6-1 testnestfor.php的输出

## 6.3 代码块和浏览器输出

在第4章，你已经学习了使用PHP开始和结束标记任意切换HTML格式。在本章中，你已经发现，根据用if和switch语句控制的判断处理过程，我们可以对用户呈现不同的输出。在本节，我们将结合这两种技术。

想象一个脚本，只有当把一个变量设为布尔值true时，它才输出一个表格的值。程序清单6.13展示了用if语句的代码块构建的一个简化的HTML表格。

程序清单6.13 包含多个echo语句的代码块

```
1: <?php
2: $display_prices = true;
3: if ($display_prices) {
4:     echo "<table border=\"1\">\n";
5:     echo "<tr><td colspan=\"3\">";
6:     echo "today's prices in dollars";
7:     echo "</td></tr>";
8:     echo "<tr><td>\$14.00</td><td>\$32.00</td><td>\$71.00</td></tr>\n";
9:     echo "</table>";
10: }
11: ?>
```

### 注意：

你要注意第8行的美元符号，当它代表字面意思并且不作为变量声明的一部分时，必须用反斜杠让它转义，把它解释成美元符号字符。如果\$display\_prices的值在第2行设为true，那么显示表格。出于可读性的考虑，我们把输出拆分到多个echo()语句中，并且再一次使用反斜杠来转义所有用于HTML输出的引号。

把上述代码放到名为testmultiecho.php的文本文件中，并把文件放到

Web服务器文档根目录下。当用Web浏览器访问这个脚本时，产生如图6-2所示的输出。

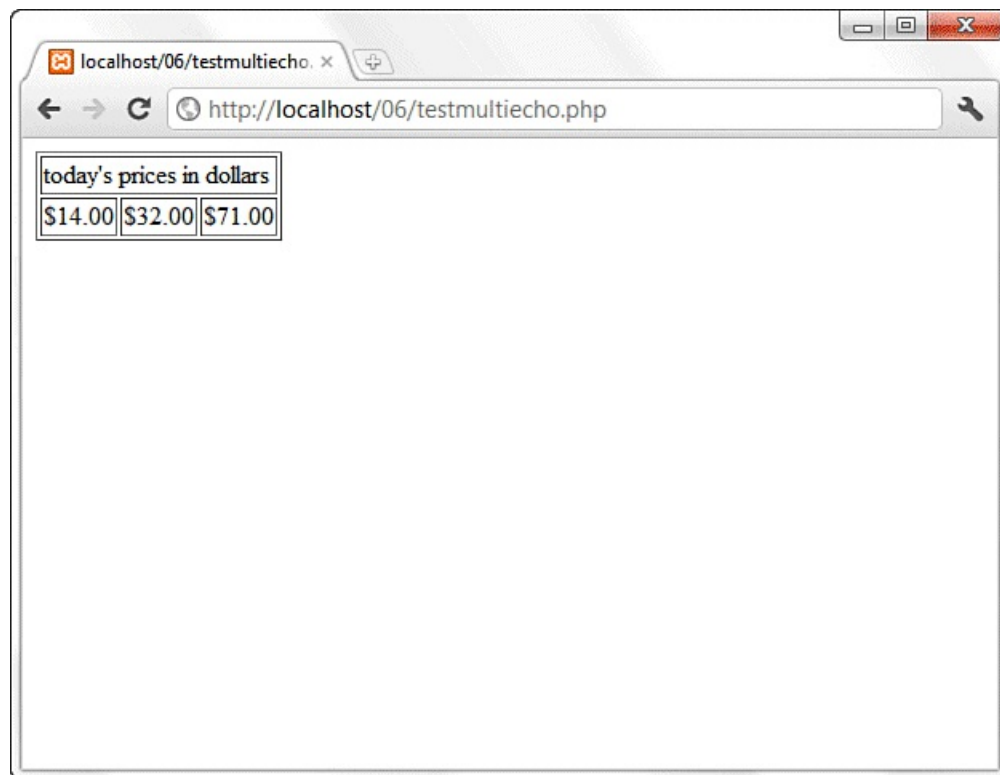


图6-2 testmultiecho.php的输出

用这种编码方法没有任何错，但通过在代码块中回到HTML模式，我们就可以省去某些输入。在程序清单6.14中我们就是这样做的。

程序清单6.14 返回代码块中的HTML模式

---

```
1: <?php
2: $display_prices = true;
3: if ($display_prices) {
4: ?>
5: <table border="1">
6: <tr><td colspan="3">today's prices in dollars</td></tr>
7: <tr><td>$14.00</td><td>$32.00</td><td>$71.00</td></tr>
8: </table>
9: <?php
10: }
11: ?>
```

---

这里值得注意的重要事情是，只有if语句的条件满足时，才会发生在第4行转换到HTML模式。这可以避免我们要转义引号和把输出放入到echo()语句中的麻烦。然而，长远来讲，这种方法可能影响代码的可读性，尤其是在脚本变得越来越大时的时候。

## 6.4 小结

在本章中，我们学习了控制结构以及能帮助我们脚本变得更灵活、更动态的方法。这些结构中的大多数将在本书后续部分中经常出现。

我们学习了如何定义if语句以及如何用elseif和else子句提供可供选择的操作。我们学习了如何根据对一个表达式的结果进行多次相等的检测，使用switch语句改变流程。我们学习了有关循环的内容，特别是while和for语句，还学习了如何使用break和continue来提前结束循环的执行或跳过一次迭代。我们学习了如何把一个循环嵌套到另一个循环之中，并且看到了这个结构的一个典型用法。最后，我们看到了一种使用PHP的开始和结束标记与条件代码块协作的技术，从而不必再对诸如引号和美元符号等特殊字符转义（在前面使用反斜杠）。

我们现在应该学会了足够的基础来自己编写做判断和执行重复任务的脚本。在下一章，我们将看到向应用程序中添加更多能力的一种新方法。将学习如何在函数中组织代码，以防止复制和提高可重用性。

## 6.5 Q&A

**Q:** 控制结构的测试表达式必然产生一个布尔值吗？

**A:** 基本上是的，但在测试表达式的上下文中，把零、一个未定义的变量或者一个空字符串转换为false。所有的其他值将计算为true。

**Q:** 控制语句中代码块必须总是使用方括号括起来吗？

**A:** 如果想要执行的代码作为控制结构的一部分，且这个控制结构仅由一个单独的行组成，那么可以省略方括号。然而，不管结构的长度有多长，总是使用开始和结束的方括号是一个好习惯。

## 6.6 实践练习

实践练习是设计用来帮助你预料可能的问题、复习已经学过的知识，并且开始把知识用于实践。



## 练习题

1. 如果一个整数变量`$age`介于18到35之间，你将如何使用if语句来向浏览器输出字符串“`Youth message`”？如果`$age`包含任何其他值，字符串“`Generic message`”将显示在浏览器上。
2. 如果变量`$age`介于1到17之间，你将如何扩展问题1中的代码以输出字符串“`Child message`”？
3. 如何创建一个在1到49之间递增并输出每一个奇数的while语句？
4. 如何将问题3中所建立的while语句转换为一个for语句？

## 解答

1.

```
$age = 22;

if (($age >= 18) && ($age <= 35)) {
    echo "Youth message";
} else {
    echo "Generic message";
}
```

2.

```
$age = 12;

if (($age >= 18) && ($age <= 35)) {
    echo "Youth message";
} elseif (($age >= 1) && ($age <= 17)) {
    echo "Child message";
} else {
    echo "Generic message";
}
```

3.

```
$num = 1;

while ($num <= 49) {
    echo $num."<br />";
    $num += 2;
}
```

4.

```
for ($num = 1; $num <= 49; $num += 2) {
    echo $num."<br />";
}
```

## 思考题

1. 回顾控制结构的语法。思考所学过的技术如何在脚本中给你带来帮助。或许某些开发脚本的思路将可以根据用户的输入表现出不同的行为，或者通过循环显示一个HTML表格。
2. 开始构建你将要使用的控制结构。暂时使用临时变量来模拟用户输入或数据库查询。

## 第7章 使用函数

在本章中，你将学到：

- 如何在脚本中定义和调用函数。
- 如何给函数传递值，以及接受由此返回的值。
- 如何使用存储在变量中的一个字符串来动态地调用一个函数。
- 如何在函数中访问全局变量。
- 如何给函数一个“内存”。
- 如何通过引用把数据传递给函数。
- 如何在调用前验证一个函数存在。

对于一个组织良好的脚本来说，函数是其核心，并且函数使得代码更容易阅读和复用。如果没有函数，大的项目就没法管理，因为，重复性代码的问题将会使开发过程陷入困境。在本章中，我们将研究函数，并且展示函数可以使你避免重复性工作的一些方法。

## 7.1 什么是函数

我们可以把函数看作是一个输入/输出的机器。这个机器接受你喂（输入）给它的原材料并且用这些原材料生产出一个产品（输出）。一个函数接受值并处理它们，然后执行一个操作（例如，显示到浏览器），或者返回一个新值，或者既执行操作又返回值。

如果你需要烘焙一个蛋糕，你可能自己做它，在自己的厨房里使用标准烤炉。但是，如果你需要烘焙上千个蛋糕，可能需要制造或购买一台专门的蛋糕烘焙机，从而保证大量地烘焙蛋糕。类似地，当决定是否创建一个函数以便重用的时候，需要考虑的最重要的因素是它可以节省你编写重复性代码的程度。

函数（**function**）是一个自包含的代码块，可以由脚本调用。当调用的时候，就执行函数的代码来完成一个特定的任务。可以向一个函数传递值，函数就会适当地使用这些值，存储它们、改变它们、显示它们，或者做任何告诉函数要去做的事情。完成以后，函数也可以向调用它的最初的代码返回一个值。

## 7.2 调用函数

函数分为两类，内建于语言中的函数和自己定义的函数。PHP有数百个内建的函数。看看下面的代码给出的使用函数的一个例子。

```
strtoupper("Hello Web!");
```

这个例子调用了`strtoupper()`函数，把字符串“Hello Web!”传递给它。然后这个函数负责自己的事务，把该字符串的内容更改为大写字母。函数调用由函数名（在这个例子中是`strtoupper`）后面跟着一个括号组成。如果想要向函数传递信息，可以把它放到括号之间。按照这种方法传递给函数的一段信息叫做参数（`argument`）。某些函数需要多个参数传递给它们，这些参数用逗号隔开，示例如下。

```
some_function($an_argument, $another_argument);
```

`strtoupper()`函数是一个典型的函数，因为它返回一个值。大多数函数在完成自己的任务之后就返回某些信息，它们通常至少告知自己的任务是否成功执行。`strtoupper()`函数返回一个字符串值，因此它的用法需要使用一个变量来接受这个新的字符串，示例如下。

```
$new_string = strtoupper("Hello Web!");
```

现在，我们可以在代码中使用`$new_string`，例如在屏幕上显示它。

```
echo $new_string;
```

这行代码将会使得如下的文本显示于屏幕上。

```
HELLO WEB!
```

提示：

`print()`和`echo()`函数并不是真正的函数，它们是用来设计把字符串输出到浏览器的语言构造。然而，你会在PHP的函数列表中找到它们，分别位于<http://www.php.net/print> 和 <http://www.php.net/echo> 。这些构造在功能上相似并且可以互换地使用。使用哪个只是你的个人偏好。

例如，`abs()`函数需要一个带符号的数字值作为参数，并且返回该数字的绝对值。让我们在程序清单7.1中试一试它。

程序清单7.1 调用内建的`abs()`函数

```
1: <?php
2: $num = -321;
3: $newnum = abs($num);
4: echo $newnum;
5: //prints "321"
6: ?>
```

在这个例子中，我们把值-321赋给一个变量`$num`。然后，我们把这个变量传递给`abs()`函数，该函数进行必要的计算并返回一个新值，我们把这个新值赋值给变量`$newnum`，然后显示结果。

把上述代码放入到一个名为`abs.php`的文本文件中，并将该文件放置在Web服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它产生如下结果。

321

实际上，我们完全可以不用临时变量，而是把我们的数值直接传递给`abs()`函数并且直接显示结果。

```
echo abs(-321);
```

然而，我们还是使用了临时变量\$num和\$newnum，从而使这个过程的每一步都尽可能的清楚。有时候，我们可以通过把代码分解成很多的简单表达式从而使其可读性更好。

可以按照和我们调用内建函数完全相同的方式来调用用户定义的函数。



## 7.3 定义一个函数

我们可以使用`function`语句来定义自己的函数。

```
function some_function($argument1, $argument2)
{
    //function code here
}
```

函数的名字跟在`function`关键词的后面，并且后面有一对括号。如果函数需要参数，必须把逗号分隔开的变量名放置在括号中，这些变量将会在函数调用时由传递给函数的值填充。即便函数不需要参数，也必须提供括号。

### 提示：

函数的命名规则和我们在第5章所学习的变量的命名规则类似。名字不能包含空格，它们必须以字母或下划线开头。和变量一样，函数名应该有意义并且风格一致。函数名大小写就是可以给你自己的代码添加的一种风格，在名字中使用混合大小写，例如`myFunction()`或`handleSomeDifficultTask()`，会使你的代码更容易阅读。你可能听说过这种叫做驼驼命令法或小写驼驼命令法的命令规则，二者的差别在于第一个字母是否小写。

程序清单7.2声明并调用了一个函数。

程序清单7.2 声明并调用一个函数

```
1: <?php
2: function bighello()
3: {
4:     echo "<h1>HELLO!</h1>";
5: }
6: bighello();
7: ?>
```

程序清单7.2中的脚本输出了包含在一个HTML的h1元素中的字符串“HELLO!”。

把上述代码放入到一个名为 `bighello.php` 的文本文件中，并且将这个文件放置在 Web 服务器文档根目录下。当通过 Web 浏览器访问这个脚本的时候，应该看到如图7-1所示的结果。



图7-1 `bighello.php`的输出

在程序清单7.2中，我们声明了一个函数**`bighello()`**，它不需要参数，因此，我们让括号保持空白。尽管**`bighello()`**是一个有效的函数，但它并不是非常有用。程序清单7.3创建了一个函数，这个函数需要一个参数，并且实际地用它做一些事情。

程序清单7.3 声明需要一个参数的函数

```
1: <?php
2: function printBR($txt)
3: {
4:     echo $txt."<br/>";
5: }
6: printBR("This is a line.");
7: printBR("This is a new line.");
8: printBR("This is yet another line.");
9: ?>
```

**提示:**

和变量名不同，函数名是不区分大小写的。在前面的例子中，`printBR()`函数也可以叫做`printbr()`、`PRINTBR()`或者任意大小写组合，都能够成功调用该函数。

把上述代码放入到一个名为`printbr.php`的文本文件中，并且将这个文件放置到Web服务器文档根目录下。当通过Web浏览器访问这个脚本的时候，应该看到如图7-2所示的结果。

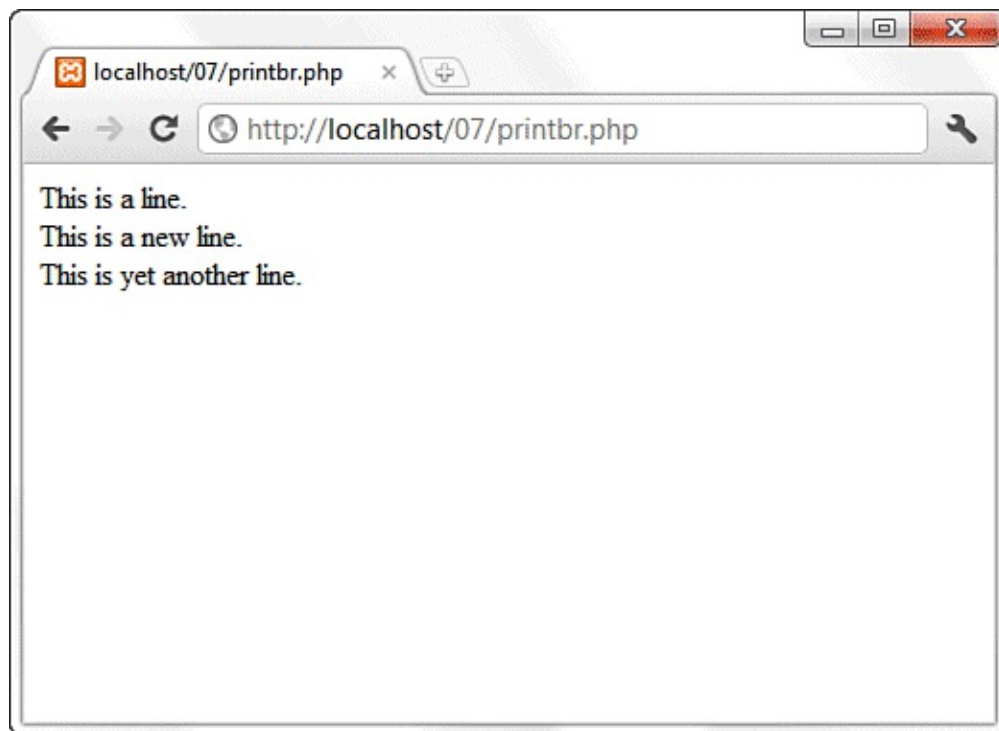


图7-2 显示一个字符串以及一个附加的<br/>标记的函数

在第2行，`printBR()`函数期待一个字符串，因此，当我们声明该函数的时候我们在括号中放入一个变量名`$txt`。不管传递给`printBR()`的是什么，都将存储到`$txt`变量中。在这个函数体中，在第3行，我们显示出`$txt`变量，并为其添加一个`<br/>`元素。

当我们想要向浏览器显示一行的时候，例如在第5行、第6行和第7行，我们可以调用`printBR()`而不是内建的`print()`，这就为我们省去了输入`<br/>`元素的麻烦。

## 7.4 从用户定义的函数返回值

在前面的例子中，我们在`printBR()`函数中向浏览器输出了一个修改后的字符串。然而，有时候，我们需要一个函数来提供一个可供自己使用的值。如果我们的函数已经修改了我们所提供的一个字符串，我们可能想要获得修改后的字符串，以便能够把它传递给其他的函数。函数可以使用`return`语句连接一个值，从而返回一个值。`return`语句停止了当前函数的执行并且把值返回给调用函数的代码。

程序清单7.4创建了一个返回两个数字的值的函数。

程序清单7.4 返回一个值的函数

---

```
1: <?php
2: function addNums($firstnum, $secondnum)
3: {
4:     $result = $firstnum + $secondnum;
5:     return $result;
6: }
7: echo addNums(3,5);
8: //will print "8"
9: ?>
```

---

把上述代码放入到一个名为`addnums.php`的文本文件中，并且将这个文件放置在Web服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它产生如下结果。

8

注意，在第2行，`addNums()`应该带有两个数值参数一起调用（在这个例子中，第6行显示这两个参数分别是3和5）。这些值存储在`$firstnum`和`$secondnum`变量中。正如预期的那样，`addNums()`把包含在

这些变量中的数字加起来并且把结果存储到名为\$result的变量中。

return语句可以返回一个值或者什么也不返回。而如何得到return语句所返回的值则各有不同。这个值可以如下直接编码。

```
return 4;
```

它也可以是一个表达式的结果，如下所示。

```
return $a/$b;
```

它还可以是另一个函数调用所返回的值，如下所示。

```
return another_function($an_argument);
```

## 7.5 变量作用域

函数中的变量声明对于该函数来说是局部的。换句话说，它在函数的外部或者在其他函数中是不可用的。在较大的项目中，当你在不同的函数中声明两个同名的变量时，这样做会防止意外地覆盖一个变量的内容。

程序清单7.5在一个函数中创建了一个变量，然后尝试在这个函数之外显示它。

程序清单7.5 变量作用域：在一个函数中声明的变量在函数外是不可用的

---

```
1: <?php
2: function test()
3: {
4:     $testvariable = "this is a test variable";
5: }
6: echo "test variable: ".$testvariable."<br/>";
7: ?>
```

---

把上述代码放入到一个名为scopetest.php的文本文件中，并且将这个文件放置在Web服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它产生如图7-3所示的结果。

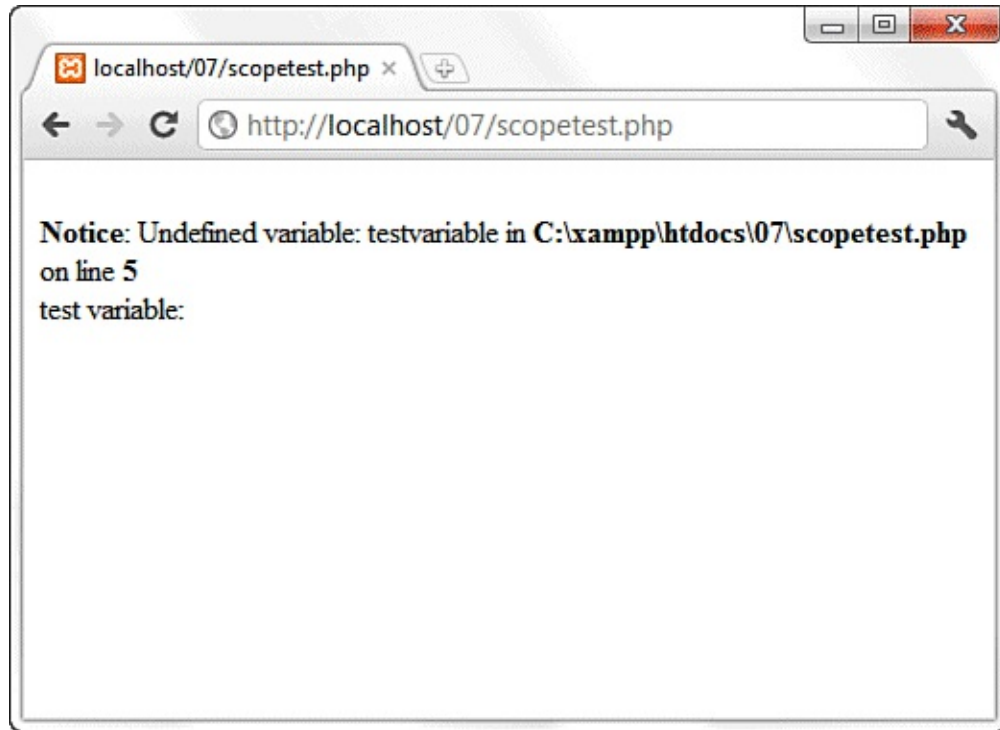


图7-3 scopetest.php的输出

提示：

你所看到的具体输出取决于PHP错误设置。也就是说，可能会也可能不会产生如图7-3所示的“notice”，但是，它会示意在“test variable”的后面缺少一个附加的字符串。

变量\$testvariable的值没有显示，因为在test()函数之外，这个变量是不存在的。别忘了，在第5行试图访问一个不存在的变量，只有PHP设置为显示所有的错误、提示和警告的时候，才会产生一个如图7-3所示的提示。如果你的错误设置是没有严格设置，将只会显示字符串“test variable:”。

同样地，在函数外声明的变量将不能自动在函数中使用。

## 使用global语句访问变量



从一个函数内部，我们不能默认地访问在另一个函数中或在脚本中其他地方定义的变量。在一个函数内部，如果试图使用具有相同名字的一个变量，你只能设置或访问局部变量。让我们在程序清单7.6中验证这一点。

程序清单7.6 默认情况下，在函数之外定义的变量不能在函数中访问

```
1: <?php
2: $life = 42;
3: function meaningOfLife()
4: {
5:     echo "The meaning of life is ".$life";
6: }
7: meaningOfLife();
8: ?>
```

把上述代码放入到一个名为scopetest2.php的文本文件中，并且将这个文件放置在Web服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它产生如图7-4所示的结果。

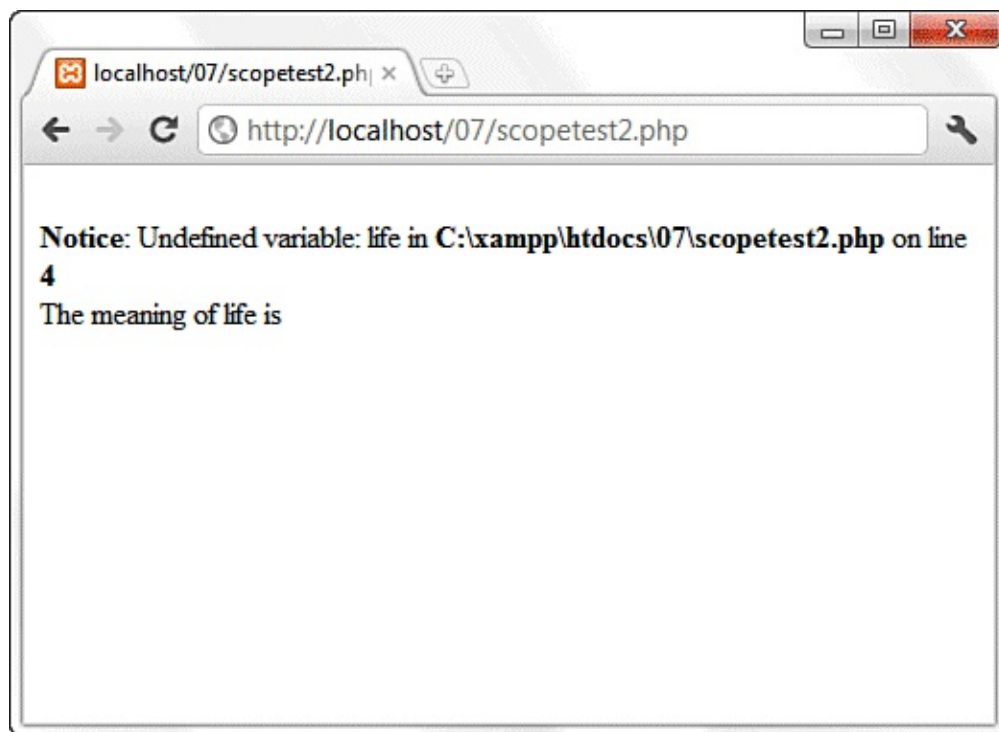


图7-4 试图在一个函数的作用域外引用一个变量

正如你所预料的，`meaningOfLife()`函数不能访问在第2行定义的`$life`变量，当函数试图显示`$life`变量的值的时候，它是空的。总的来说，这是一件好事，因为它避免了我们在同名的变量之间的潜在的冲突，并且如果一个函数需要外界的信息，它可以通过一个参数。偶尔，你可能希望在一个函数内部访问一个重要的变量，而又不想将它作为一个参数传递进来，这就是`global`语句的用武之地。程序清单7.7使用`global`来使一切恢复正常。

程序清单7.7 使用`global`语句访问全局变量

---

```
1: <?php
2: $life=42;
3: function meaningOfLife()
4: {
5:     global $life;
6:     echo "The meaning of life is ".$life";
7: }
8: meaningOfLife();
9: ?>
```

---

把上述代码放入到一个名为`scopetest3.php`的文本文件中，并且将这个文件放置在Web服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它产生如图7-5所示的结果。

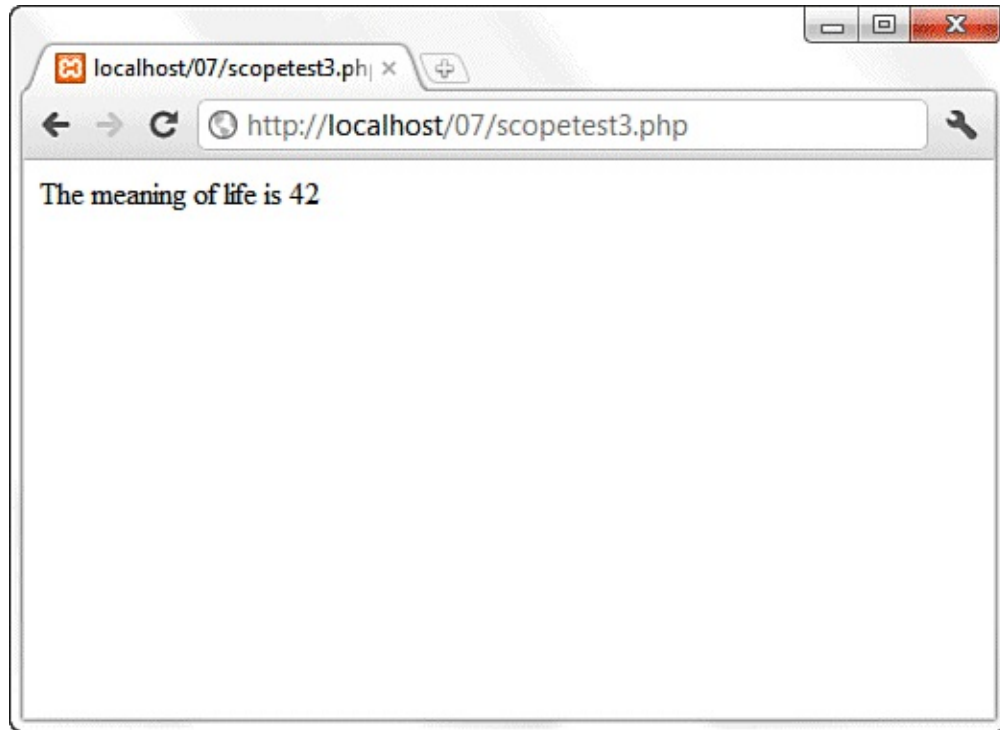


图7-5 在一个函数中，使用global语句成功地访问一个全局变量

当我们在meaningOfLife()函数中声明\$life变量的时候（第4行），通过把global语句放置在它前面，就可以引用在函数的外面（第2行）声明的\$life变量了。在需要访问一个特别指定的全局变量的函数中，都需要使用global语句。但是请留意，如果在函数中操作变量的内容，变量的值可能会在整个脚本的范围内都发生变化。

我们可以使用global语句一次声明多个变量，只需要用逗号将想要访问的每个变量隔开，示例如下。

```
global $var1, $var2, $var3;
```

**注意：**

通常，参数就是调用代码所传递的任何值的一个副本，在函数中修改它，对于函数块以外的部分没有任何影响。另一方面，在一个函数中修改一个全局变量，则会修改原始值而不

是副本。因此，使用global语句的时候要小心。

## 7.6 使用static语句在函数调用之间保存状态

函数中的局部变量拥有一个短暂而快乐的人生，它们产生于函数调用的时候，而当函数执行完成的时候它们就消亡了。

然而，偶尔我们也会需要给函数一个基本的内存。假设我们需要一个函数来记录它已经被调用的次数，以便一个脚本可以创建一个次数标题。我们当然能使用global语句做到这一点，如程序清单7.8所示。

程序清单7.8 使用global语句在函数调用之间记录一个变量的值

---

```
1: <?php
2: $num_of_calls = 0;
3: function numberedHeading($txt)
4: {
5:     global $num_of_calls;
6:     $num_of_calls++;
7:     echo "<h1>".$num_of_calls." ".$txt."</h1>";
8: }
9: numberedHeading("Widgets");
10: echo "<p>We build a fine range of widgets.</p>";
11: numberedHeading("Doodads");
12: echo "<p>Finest in the world.</p>";
13: ?>
```

---

把上述代码放入到一个名为numberedheading.php的文本文件中，并且将这个文件放在Web服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它将产生如图7-6所示的结果。

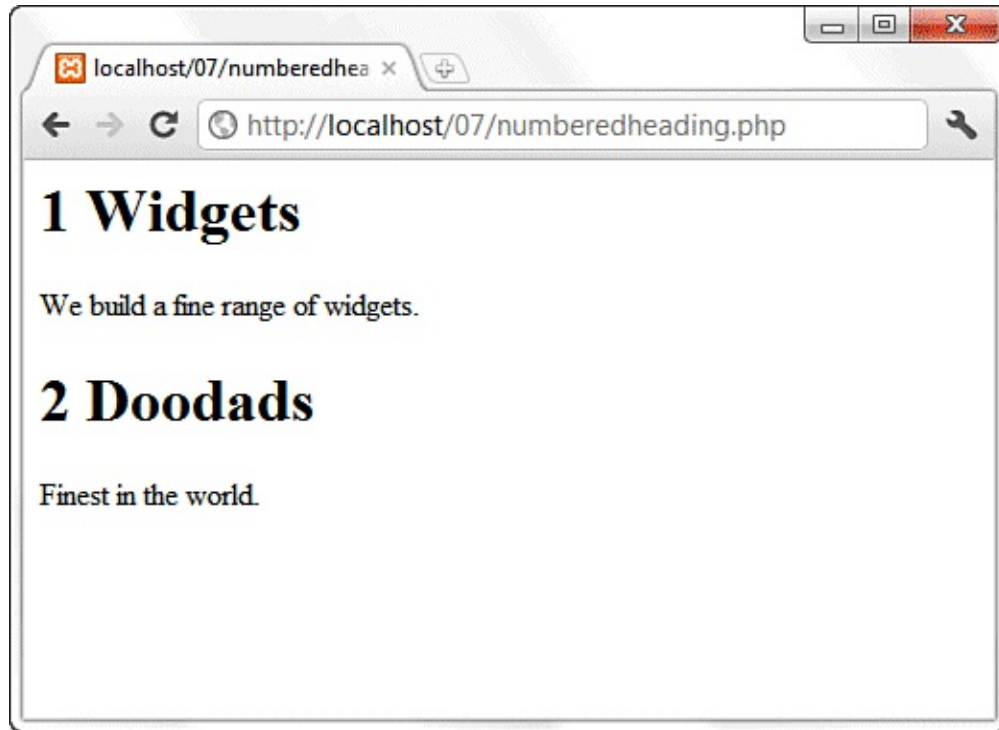


图7-6 使用global语句来记录一个函数调用的次数

这确实有效。我们在`numberedHeading()`函数的外部，即第2行声明了一个变量`$num_of_calls`。在第5行使用一条`global`语句使这个变量变得对函数可用。

每次调用`numberedHeading()`，`$num_of_calls`的值都会增加1（在第6行），然后我们可以使用正确的、递增后的标题编号来显示这个标题。

然而，这还不是最优雅解决方案。使用`global`语句的函数不能作为独立的代码段阅读。在阅读或复用这些函数的时候，我们需要小心它们所操作的全局变量。

这种情况下，`static`语句就派上用场了。如果你在一个函数中使用`static`语句声明了一个变量，这个变量对于该函数仍保持为局部的，并且函数在从一次执行到另一次执行的过程中会“记住”该变量的值。程序清

单7.9修改了程序清单7.8的代码，并且使用了static语句。

程序清单7.9 使用static语句在函数调用之间记住一个变量的值

---

```
1: <?php
2: function numberedHeading($txt)
3: {
4:     static $num_of_calls = 0;
5:     $num_of_calls++;
6:     echo "<h1>".$num_of_calls." ". $txt."</h1>";
7: }
8: numberedHeading("Widgets");
9: echo "<p>We build a fine range of widgets.</p>";
10: numberedHeading("Doodads");
11: echo "<p>Finest in the world.</p>";
12: ?>
```

---

numberedHeading()函数已经变得完全自包含了。当在第4行声明\$num\_of\_calls变量时，我们就把初始值赋给了它。当函数在第8行第一次调用的时候，这个赋值就进行了。当函数在第10行第二次调用的时候，这个最初的赋值被忽略了，相反，代码记住了\$num\_of\_calls的前一个值。我们现在可以把numberedHeading()函数粘贴到其他的脚本中，而不需要担心全局变量。尽管程序清单7.9的输出确实和程序清单7.8相同，但我们使代码更为优雅。

## 7.7 关于参数的更多内容

我们已经看到了如何把参数传递给函数，但是这还不够。在本节中，我们将会看到一种给参数设置默认值的技术，并且找到一种通过引用而不是通过值来传递参数值的方法。传递引用意味着，给了函数参数一个最初值的别名而不是它的一个副本。

### 7.7.1 为参数设置默认值

PHP提供了一个极好的功能帮助构建灵活的函数。到目前为止，我们已经提到一些函数需要一个或多个参数，通过使某些参数变为可选的，我们可以让函数少一些专有性。

程序清单7.10创建了一个有用的小函数，它把一个字符串包含在一个HTML的span元素中。我们想要给函数的用户一个能够修改font-size样式的机会，因此，除了字符串之外我们还需要一个参数\$fontsize（第2行）。

程序清单7.10 需要两个参数的一个函数

---

```
1: <?php
2: function fontWrap($txt, $fontsize)
3: {
4:     echo "<span style=\"font-size:$fontsize\">".$txt."</span>";
5: }
6: fontWrap("A Heading<br/>", "24pt");
7: fontWrap("some body text<br/>", "16pt");
8: fontWrap("smaller body text<br/>", "12pt");
9: fontWrap("even smaller body text<br/>", "10pt");
10: ?>
```

---

把上述代码放入到一个名为fontwrap.php的文本文件中，并且将这



个文件放置在Web服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它将产生如图7-7所示的结果。



图7-7 格式化并输出字符串的一个函数

通过在函数定义的括号中把一个值赋给一个参数变量，我们可以使\$fontsize参数变为可选的。如果函数调用时没有为这个参数传递一个参数值，我们定义时赋给参数的值就将被采用。程序清单7.11使用这种技术来使得\$fontsize参数变为可选的。

程序清单7.11 带有一个可选参数的函数

```
1: <?php
2: function fontWrap($txt, $fontsize = "12pt")
3: {
4:     echo "<span style=\"font-size:$fontsize\">".$txt."</span>";
```

---

```
5: }
6: fontWrap("A Heading<br/>", "24pt");
7: fontWrap("some body text<br/>");
8: fontWrap("smaller body text<br/>");
9: fontWrap("even smaller body text<br/>");
10: ?>
```

---

当使用两个参数调用fontWrap()函数的时候，如第6行所示，第2个参数值用来设置span元素的font-size属性。当我们忽略了这个参数，如第7行、第8行和第9行所示，就使用默认值“12pt”。可以创建任意多个可选参数，但是当我们给一个可选参数赋一个默认值时，所有后续的参数也都应该给定默认值。

## 7.7.2 把变量引用传递给函数

当我们把参数传递给函数时，它们作为副本存储在参数变量中。在函数体中对这些变量的任何修改都是局部的，并且不会反映到函数之外，这一点可以通过程序清单7.12来说明。

程序清单7.12 通过值来把参数传递给一个函数

---

```
1: <?php
2: function addFive($num)
3: {
4:     $num += 5;
5: }
6: $orignum = 10;
7: addFive($orignum);
8: echo $orignum;
9: ?>
```

---

把上述代码放入到一个名为addfive.php的文本文件中，并且将这个文件放置在 Web 服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它将产生如下结果。

`addFive()`函数接受一个单独的数值并且将它加上5，但是它不返回什么。在第6行我们把一个值赋给一个变量`$orignum`，然后在第7行把这个变量传递给`addFive()`函数，`$orignum`内容的一个副本存储在变量`$num`中，尽管我们把`$num`增加了5，但这对于`$orignum`的值还是没有影响。当我们显示`$orignum`，会看到其值仍然是10。默认情况下，传递给函数的变量是根据值来传递的。换句话说，制造了变量的值的本地副本。

我们可以通过创建一个对初始变量的引用来改变这种行为，可以把一个引用理解成指向变量的一个路标。在使用引用的时候，我们操作它所指向的变量的值。

程序清单7.13展示了这一技术。当我们通过引用把一个参数传递给一个函数的时候，如第7行所示，我们所传递的变量（`$orignum`）的内容在函数中被这个参数变量访问并操作，而不只是变量的值的一个副本（10）被访问和操作。在这些情况下，对一个参数的任何改变都会改变最初变量的值。我们可以通过在函数定义中的参数名的前面添加一个`&`符号，从而用引用来传递一个变量，如第2行所示。

程序清单7.13 使用一个函数定义，把参数通过引用传递给函数

---

```
1: <?php
2: function addFive(&$num)
3: {
4:     $num += 5;
5: }
6: $orignum = 10;
7: addFive($orignum);
8: echo $orignum;
9: ?>
```

---

把上述代码放入到一个名为`addfive2.php`的文本文件中，并且将这个文件放置在Web服务器文档根目录下。当通过Web浏览器来访问这个

脚本的时候，它将产生如下结果。

## 7.8 测试函数是否存在

在尝试调用函数之前，我们未必总是知道该函数是否存在。PHP引擎的不同构建可能包含不同的功能，如果所编写的脚本可以在多个服务器上运行，我们可能需要验证关键功能是否可用。例如，你可能想要编写这样的代码：如果MySQL相关的功能可以使用的话就使用MySQL，否则只是简单地把数据记录到一个文本文件中。

可以使用`function_exists()`函数来检查函数的可用性。`function_exists()`需要一个表示函数名的字符串，如果可以找到该函数，它返回`true`，否则返回`false`。

程序清单7.14示意了`function_exists()`函数的使用，并且说明了我们在本章中介绍到的其他的一些主题。

程序清单7.14 测试一个函数的存在性

---

```
1: <?php
2: function tagWrap($tag, $txt, $func = "")
3: {
4:     if ((!empty($txt)) && (function_exists($func))) {
5:         $txt = $func($txt);
6:         return "<". $tag. ">". $txt. "</". $tag. "><br/>";
7:     } else {
8:         return "<strong>". $txt. "</strong><br/>";
9:     }
10: }
11:
12: function underline($txt)
13: {
14:     return "<span style=\"text-decoration:underline;\">". $txt. "</span>";
15: }
16: echo tagWrap('strong', 'make me bold');
17: echo tagWrap('em', 'underline and italicize me', "underline");
18: echo tagWrap('em', 'make me italic and quote me',
19: create_function('$txt', 'return "&quot;$txt&quot;";'));
20: ?>
```

---

我们定义了两个函数，tagWrap()（在第2行）和underline()（在第12行）。tagWrap()函数接受3个内容：一个标记、要格式化的文本以及一个可选的函数名，它返回一个格式化后的字符串。underline()函数需要一个参数，即要格式化的文本，并且返回包含在<span>标记中的文本，而<span>标记带有相应的样式属性。

当在第16行第一次调用tagWrap()的时候，我们把字符 'strong' 和字符串“make me bold”传递给它。由于我们没有给函数参数传递一个值，就使用默认的值（一个空字符串）。在第4行，我们检查\$txt变量是否包含字符串，以及\$func是否存在，我们调用function\_exists()来根据这个名字检查一个函数。当然，在这个例子中，\$func变量是空的，因此，我们在第7到8行的else子句中把\$txt变量包含到<strong>标记中并返回结果。

我们在第17行使用字符串‘em’、某些文本以及第三个参

数“underline”调用tagWrap()。function\_exists()查找名为underline()的函数（在第12行），于是它调用这个函数，并且在进行任何进一步的格式化之前把参数变量\$txt传递给它。结果是一个斜体、带下划线的字符串。

最后，在第18行，我们调用tagWrap()，它把文本放到引用实体中。直接给要改变的文本添加实体，这应该更快，但是，这个示例主要用来说明一点，即function\_exists()函数对于匿名函数和对于表示函数名的字符串一样有效。

把上述代码放入到一个名为exists.php的文本文件中，并且将这个文件放置在Web服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它将产生如图7-8所示的结果。

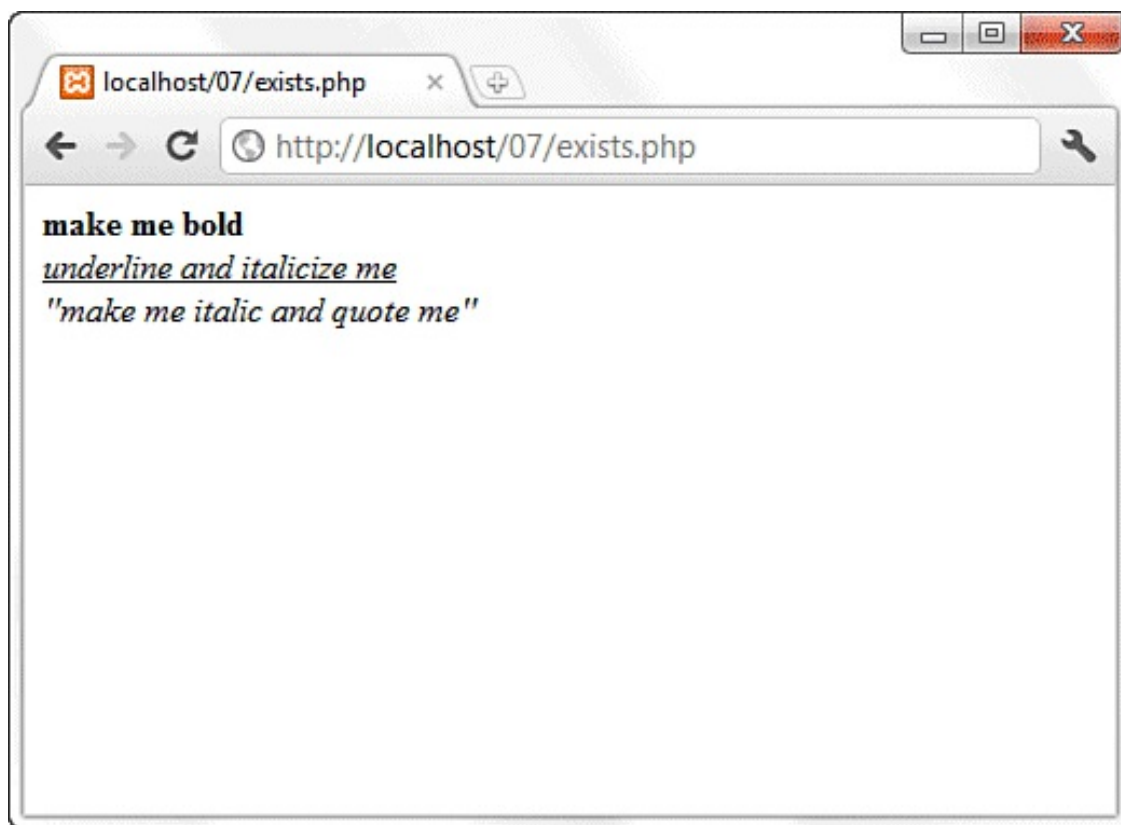


图7-8 exists.php的输出

## 7.9 小结

本章介绍了有关函数的知识，以及如何使用它们。我们学习了如何定义函数并向函数传递参数，如何使用`global`和`static`语句，如何向函数传递引用，以及如何为函数参数创建默认值。最后，我们学习了测试函数的存在性。



## 7.10 Q&A

**Q:** 可以在一个双引号或单引号字符串中包含一个函数调用吗，就像我们对一个变量所做的那样？

**A:** 不可以。我们必须在引号外调用函数。然而，我们可以把一个字符串分开，并且把函数调用放在字符串之间，使用连接操作符把它们连接起来，示例如下。

```
$newstring = "I purchased".numPurchase($somenum)." items.";
```

**Q:** 如果我们调用一个不存在的函数，或者我们用一个已经使用的名字来声明一个函数，会发生什么情况？

**A:** 调用一个不存在的函数或者使用和其他已有的函数同样的名字来声明一个函数，将会导致脚本停止执行。浏览器中是否显示一条错误消息，取决于你的php.ini文件中的错误设置。

## 7.11 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 判断对错：如果一个函数不需要一个参数，你可以在函数调用中省略括号。

2. 如何从函数返回一个值？

3. 如下代码段将向浏览器显示什么？

```
$number = 50;  
  
function tenTimes() {  
    $number = $number * 10;  
}  
  
tenTimes();  
echo $number;
```

4. 如下代码段将向浏览器显示什么？

```
$number = 50;  
  
function tenTimes() {  
    global $number;  
    $number = $number * 10;  
}  
  
tenTimes();  
echo $number;
```

5. 如下代码段将向浏览器显示什么？

```
$number = 50;

function tenTimes( &$n ) {
    $n = $n * 10;
}

tenTimes( $number );
echo $number;
```

## 解答

1. 这句话是错的。我们必须总是在函数调用中包含括号，无论是否向函数传递参数。
2. 必须使用`return`关键字。
3. 它显示50。`tenTimes()`函数没有访问全局变量`$number`。当调用该函数的时候，它将会操作自己的局部变量`$number`。
4. 它显示500。我们使用了`global`语句，这会让`tenTimes()`函数访问外部`$number`变量。
5. 它显示500。通过向参数变量`$n`添加`&`符号，我们确保了参数按引用传递。`$n`和`$number`指向同一个值，因此，对`$n`的任何改变都将在`$number`反映出来。

## 思考题

1. 回顾创建一个函数的语法，该函数接受参数、操作这些值，并且返回一个字符串。
2. 创建一个函数，它接受4个字符串变量并且返回一个字符串，其中包含了一个HTML表格元素，把每个变量都放置在自己的单元格中。

## 第8章 使用数组

在本章，我们将学习：

- 如何创建关联数组和多维数组。
- 如何使用**PHP**内建的众多和数组相关的函数。

数组用来存储和组织数据。**PHP**包括很多与数组相关的函数，这些函数使得你可以创建、修改和操作数组，而数组将在本书所描述的过程式程序设计方法中频繁地使用。

## 8.1 什么是数组

在本书前面的各章，我们已经学习并使用了标量变量，并且知道这些变量用来存储值。但是，标量变量一次只能存储一个值，如`$color`变量只能存储一个`red`值或`blue`值等，但它无法用来存储彩虹中的颜色列表。而数组是一种特殊类型的变量，它使得我们能够存储任意多个值，包括彩虹中的所有7种颜色。

尽管可以在一个数组中存储尽可能多的值，但一些数组函数还是有100000个值的上限。如果在数组中要存储大量的数据，确保阅读PHP手册中有关你要使用的数组函数的条目，看看该函数是否有操作数据的上限。

数组是有索引的，这意味着每个条目都由一个键（`key`）和一个值（`value`）组成。键是索引的位置，从0开始并且对于数组中的每个新元素都增加1。值就是我们和该位置关联起来的任何值：一个字符串、一个整数或任何我们希望的。可以把一个数组看作是一个文件柜，而一个键/值对就是一个文件夹，键就是文件夹上面的标签，而值就是文件夹中的文件。当我们在下一节中创建数组的时候，就会看到这种类型的结构的实际应用。



## 8.2 创建数组

我们可以使用`array()`函数或者数组操作符`[]`来创建一个数组。当我们想要一次性创建一个新的数组并且用多个元素来填充它的时候，通常使用`array()`函数。当我们想要创建一个新的数组并且它开始的时候只有一个元素，或者当我们想要添加一个已经存在的数组元素的时候，通常使用数组操作符。

如下的代码段显示了如何使用`array()`函数创建一个名为`$rainbow`的数组，其中包含了彩虹所有颜色。

```
$rainbow = array("red", "orange", "yellow", "green", "blue", "indigo",  
    "violet");
```

如下的代码段展示了使用数组操作符来逐渐地创建同一个数组的方法。

```
$rainbow[] = "red";  
$rainbow[] = "orange";  
$rainbow[] = "yellow";  
$rainbow[] = "green";  
$rainbow[] = "blue";  
$rainbow[] = "indigo";  
$rainbow[] = "violet";
```

这两个代码段都创建了一个名为`$rainbow`的7元素的数组，其值从索引位置0开始到索引位置6结束。如果你想要排列出它们，可以指定索引位置，像下面这样编写代码。

```
$rainbow[0] = "red";  
$rainbow[1] = "orange";  
$rainbow[2] = "yellow";
```

```
$rainbow[3] = "green";  
$rainbow[4] = "blue";  
$rainbow[5] = "indigo";  
$rainbow[6] = "violet";
```

然而，没有指定位置的时候，PHP会为你做这些，并且会消除像下面的例子这样排错元素的可能性。

```
$rainbow[0] = "red";  
$rainbow[1] = "orange";  
$rainbow[2] = "yellow";  
$rainbow[5] = "green";  
$rainbow[6] = "blue";  
$rainbow[7] = "indigo";  
$rainbow[8] = "violet";
```

不管最初是使用`array()`函数还是数组操作符创建数组，你都可以使用数组操作符来添加数组。在下面的第一行中，6个元素添加到了数组中，在第二行，另外一个元素添加到了数组的末尾。

```
$rainbow = array("red", "orange", "yellow", "green", "blue", "indigo");  
$rainbow[] = "violet";
```

本节中使用的数组是数字索引的数组，也是最为常见的数组类型。在下面两节中，我们将学习两种其他类型的数组：关联数组和多维数组。

### 8.2.1 创建关联数组

数字索引数组使用一个索引位置作为键，如0、1、2等，而关联数组使用实际命名的键。下面的例子通过创建一个具有4个元素的名为`$character`的数组来说明这一点。

```
$character = array(  
    "name" => "Bob",  
    "occupation" => "superhero",  
    "age" => 30,  
    "special power" => "x-ray vision"  
);
```

\$character数组中的4个键是name、occupation、age和special power。关联的值分别是Bob、superhero、30和x-ray vision。我们可以使用指定的键来引用关联数组的具体元素，如下面的例子所示。

```
echo $character['occupation'];
```

上述代码段的输出如下。

```
superhero
```

和数字索引数组一样，我们可以使用数组操作符来添加一个关联数组，示例如下。

```
$character['supername'] = "Mega X-Ray Guy";
```

这个例子添加了一个名为supername的键，其值为Mega X-Ray Guy。

一个关联数组和一个数字索引数组之间的唯一的区别就是键名，在一个数字索引数组中，键名是数字。在一个关联数组中，键名是一个有意义的单词。

## 8.2.2 创建多维数组

前面介绍的两种数组分别存储字符串和整数，而这里介绍的数组则存储其他的数组。如果每组键/值对构成了一维，一个多维数组存储了多组这样的键/值对。例如，程序清单8.1定义了一个名为\$characters的多

维数组，其每一个元素都包含一个关联数组。这听起来可能容易混淆，但是，它实际上只是包含其他数组的一个数组。

程序清单8.1 定义一个多维数组

---

```
1: <?php
2: $characters = array(
3:     array(
4:         "name" => "Bob",
5:         "occupation" => "superhero",
6:         "age" => 30,
7:         "special power" => "x-ray vision"
8:     ),
9:     array(
10:        "name" => "Sally",
11:        "occupation" => "superhero",
12:        "age" => 24,
13:        "special power" => "superhuman strength"
14:    ),
15:    array(
16:        "name" => "Jane",
17:        "occupation" => "arch villain",
18:        "age" => 45,
19:        "special power" => "nanotechnology"
20:    )
21: );
22: ?>
```

---

在第2行，使用array()函数初始化数组\$characters。第3~8行显示了\$characters数组的第一个元素，第9~14行显示了\$characters数组的第二个元素，而第15~20行显示了\$characters数组的第三个元素。这些元素可以用\$characters[0]、\$characters[1]和\$characters[2]来引用。

每个元素都由一个关联数组组成，这个关联数组自身包含4个元素：name、occupation、age和special power。

然而，如果你试图像下面这样显示主元素

```
echo $characters[1];
```

输出的结果将会如下。

#### Array

因为主元素实际上保存了一个数组作为其内容。要想真正地得到想要的元素，即在内部数组元素中包含的具体信息，我们需要访问主元素索引位置以及想要浏览的值的关联名。

来看一个例子。

```
echo $characters[1]['occupation'];
```

它将会显示如下结果。

#### superhero

如果把如下的代码添加到程序清单8.1的末尾，它将显示出存储在每个元素中的信息，并在浏览器中多显示出一行分隔线。

```
foreach ($characters as $c) {  
    while (list($k, $v) = each ($c)) {  
        echo "$k ... $v <br/>";  
    }  
    echo "<hr/>";  
}
```

foreach循环和主数组元素\$characters相关，它遍历这个数组，并且把临时变量名\$c赋给包含在每个位置中的元素。接下来，我们开始一个while循环，这个循环使用两个函数来提取内部数组的内容。首先，list()函数命名了占位符变量\$k和\$v，这两个变量将使用each()函数所收集的键和值来填充。each()函数查看\$c数组的每个元素并且相应地提取信息。

echo语句只是显示出使用each()函数从\$c数组提取的每个键和值

（\$k和\$v），并且添加一条分隔线以便于显示。图8-1展示了这个名为mdarray.php的文件的结果。

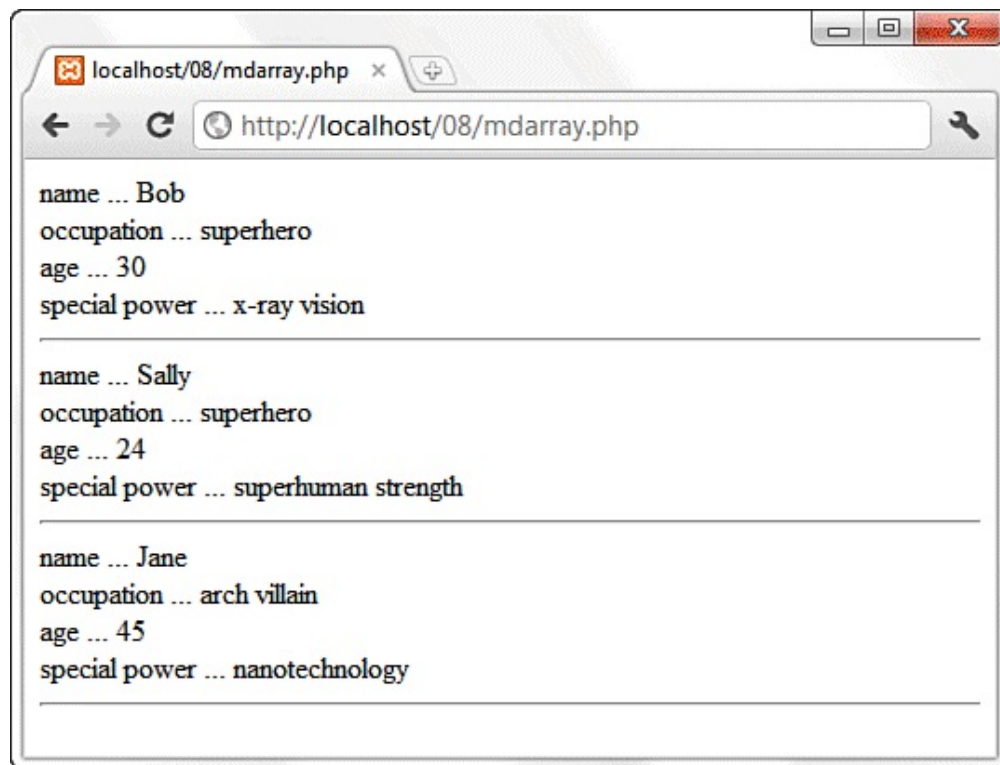


图8-1 遍历一个多维数组

## 8.3 一些和数组相关的函数

PHP内建了70多个和数组相关的函数，我们可以从<http://www.php.net/array> 了解详细信息。本节介绍其中一些很常用的，并且很有用的函数。

- `count()`和`sizeof()`——这两个函数都会计算一个数组中的元素个数。

给定如下的数组

```
$colors = array("blue", "black", "red", "green");
```

`count($colors);` 和 `sizeof($colors);` 都返回值4。

- `each()`和`list()`——在遍历一个数组并返回其键和值的应用中，这两个函数通常一起出现（`list()`是看上去像一个函数一样的语言结构）。我们在前面见到过其应用的一个例子，在那里，我们遍历了`$c`数组并显示其内容。
- `foreach()`——这个控制结构也用来遍历一个数组，把一个元素的值赋给一个给定的变量，正如我们在上一节中见到的。
- `reset()`——这个函数把指针返回到一个数组的开始，示例如下。

```
reset($character);
```

当我们要对一个数组执行多个操作的时候，例如排序、提取值等，这个函数很有用。

- `array_push()`——这个函数在一个已有数组的末尾添加一个或多个元素，示例如下。

```
array_push($existingArray, "element 1", "element 2", "element 3");
```

- `array_pop()`——这个函数删除并返回一个已有数组的最后一个元素，示例如下。

```
$last_element = array_pop($existingArray);
```

- `array_unshift()`——这个函数在一个已有数组的开头添加一个或多个元素，示例如下。

```
array_unshift($existingArray, "element 1", "element 2", "element 3");
```

- `array_shift()`——这个函数删除并返回一个已有数组的第一个元素，例如，把`$existingArray`的第一个位置的元素的值赋给变量`$first_element`，示例如下。

```
$first_element = array_shift($existingArray);
```

- `array_merge()`——这个函数组合两个或多个已有的数组，示例如下。

```
$newArray = array_merge($array1, $array2);
```

- `array_keys()`——这个函数返回一个数组，其中包含了一个给定数组中的所有键名，示例如下。

```
$keysArray = array_keys($existingArray);
```

- `array_values()`——这个函数返回一个数组，其中包含了一个给定数组中的所有值，示例如下。

```
$valuesArray = array_values($existingArray);
```

- `shuffle()`——这个函数把一个给定数组的所有元素随机排列。这个函数的语法如下。

```
shuffle($existingArray);
```

这个与数组相关的函数的简短概要对于使用数组来说只是隔靴挠痒。然而，数组以及和数组相关的函数在整本书的代码示例中都会用



到，因此，你将会很快熟悉它们。如果你做不到，位于 <http://www.php.net/array> 的PHP手册也有部分数组可供参考，它详细讨论了和数组相关的所有函数，包括排序数组的十多种不同的方法。

## 8.4 小结

本章介绍了数组的概念，包括如何创建和引用它们。3种数组类型是数字索引数组、关联数组和多维数组。此外，我们看到了PHP内建的众多数组相关函数中的一些例子。这些函数可以用来操作和修改已有的数组，有时候甚至可以创建一个全新的数组。

## 8.5 Q&A

**Q:** 多维数组可以有多少维？

**A:** 可以在多维数组中创建任意多个维，但是别忘了，维数越多，管理越难。如果你拥有多维的数据，明智的方法是看看这些数据是否可以用不同的方法存储以及是否可以用这种方法访问，例如在数据库中。

**Q:** 如果只是想要创建一个联系表单，为什么要关心数组？

**A:** 即使在最基本的客户端/服务器交互中（例如，Web站点中的一个联系表单），数组也是有用的。我们将会在第11章学习关于表单的更多知识，但是，在开始使用本章中的知识之前，有几点需要记住。如果表单包含了任何复选框或列表，用户可以从其中选择多个选项，那么，这些数据应该作为数组发送给表单。如果要操作这些数据的话，你需要将数据从数组中取出来，本章给出了几个基本的示例就是这么做的。

## 8.6 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 什么样的结构可以用来定义一个数组？
2. 什么函数可以用来连接两个数组？

## 解答

1. `array()`
2. `array_merge()`

## 思考题

1. 回顾定义一个多维数组并引用其数据项的过程。
2. 创建一个按照类型组织的电影的多维数组。它将会采用一个关联数组的形式，电影的类型如Science Fiction、Action、Adventure等作为键。这些数组元素的每一个都应该是包含电影名（例如Alien、Terminator 3、Star Wars等）的一个数组。创建了数组之后，遍历它们，显示出每种类型及其相关电影的名字。

## 第9章 使用对象

在本章中，你将学到：

- 理解一个对象的基本结构。
- 如何创建和操作对象以及它们所包含的数据。

对象用来存储和组织数据。面向对象编程是这样一种编程范型，其中程序（或应用程序）的结果围绕着这些对象以及对象之间的关系和交互来进行。很多程序设计语言中都拥有的面向对象程序设计结构，而这在PHP中也可以找到。

实际上，很多PHP程序员，尤其是那些具有较强的面向对象编程语言背景的程序员，会选择以面向对象的方式来开发PHP应用程序。

然而，在PHP中，你不一定必须用面向对象的方式来编写脚本。很多PHP脚本，实际上我们在本书中见到的大多数脚本，都是过程式的而不是面向对象的。也就是说，强调在创建程序的过程中，使用变量、数据和控制结构、例程和函数。原因很简单，如果你全新地接触一种编程，那么，在接触一种本身有很多相关的图书的编程范型之前，获取过程式编程的经验以及这种语言的基础知识是很重要的。本章带你简单地领略一下对象的世界，因为对于你继续获取更多的知识和经验来说，这是一种重要的数据类型和概念。

如果你带着在面向对象程序设计中的背景来学习PHP，本章将帮助



你理解PHP中的对象模型。

## 9.1 创建一个对象

解释一个对象的概念有点难：它是事物的一个理论化的盒子，这些事物（变量、函数等）都存在于叫做类（`class`）的一个模板式结构中。尽管很容易形象地描述一个标量变量，例如带有`red`值的`$color`，或者其中带有3个或4个不同元素的名为`$character`的一个数组，某些人还是在形象地描述对象的时候感到为难。

从现在开始，尝试把对象看作一个小盒子，其两端带有输入和输出。输入的机制叫做方法（`method`），方法具有属性（`property`）。在整个本章中，我们将看到类、方法和属性是如何协同工作产生各种输出的。

### 提示：

如果你对类的概念是完全陌生的，可以通过阅读PHP手册的“Classes and Objects”一章来充实自己的知识。可以在<http://www.php.net/manual/en/language.oop5.php> 找到它。

本节开始的时候提到，一个对象拥有一个叫做类的结构。在每个类中，我们定义了一组特征。例如，假设你已经创建了一个`automobile`类，在`automobile`类中，我们应该有`color`、`make`、`model`特征。每个`automobile`对象使用所有的这些特征，但它们都初始化为不同的值，例如`silver`、`Mazda`和`Protege5`，或者`red`、`Porsche`和`Boxter`。

使用对象的全部目标就是创建可以重用的代码。因为类是如此紧密的结构但又是自包含的，并且彼此独立，它们可以从一个应用程序重用

到另一个应用程序中。例如，假设你编写了文本格式化类用于一个项目，并且决定在另一个项目中使用这个类。由于这个类只是一组特征，你可以取出代码并且在第二个项目中使用它，使用第二个项目特定的方法来进入其中，但却使用已有代码的内部工作机制来得到新的结果。

创建一个类很简单，只用声明其存在即可，示例如下。

```
class myClass {  
    //code will go here  
}
```

既然已经有了一个类，我们可以创建对象的一个新实例，示例如下。

```
$object1 = new myClass();
```

在程序清单9.1中，你已经证实了对象的存在，即便其中没有什么内容——它现在只有一个名字。

程序清单9.1 证实对象的存在

---

```
1: <?php  
2: class myClass {  
3:     //code will go here  
4: }  
5: $object1 = new myClass();  
6: echo "\$object1 is an ".gettype($object1)."<br/>";  
7:  
8: if (is_object($object1)) {  
9:     echo "Really! I swear \$object1 is an object!";  
10: }  
11: ?>
```

---

如果将上述代码保存为proofofclass.php，将其放置在文档根目录下，使用Web浏览器访问它，将会看到如下的显示结果。

```
$object1 is an object.  
Really! I swear $object1 is an object!
```

这不是一个特别有用的类，因为它并没有做什么事情，但它是有效的，并且通过第2行到第5行展示了类模板是如何工作的。第8行到第10行使用`is_object()`函数来测试某物是否是一个对象，在这个例子中“某物”就是`$object1`。由于`is_object()`的测试结果为`true`，`if`语句中的字符串将显示到屏幕上。

接下来，我们将学习对象的属性和方法，它们增加了类模板的用途。

### 9.1.1 对象的属性

在对象中声明的变量叫做属性（`property`）。在一个类的顶部声明变量，这是标准的做法。这些属性可以是值、数组甚至是其他对象。如下的代码段在类中使用简单的标量变量，其前面有一个公共关键字。

```
class myCar {  
    public $color = "silver";  
    public $make = "Mazda";  
    public $model = "Protege5";  
}
```

如果你在变量名称前使用关键字`public`、`protected`或`private`，那么，可以表明类成员（变量）是在任意地方都可以访问（`public`）、在类自身或父类或继承类中可以访问（`protected`）、还是只能由该类自身访问（`private`）。

现在，当你创建一个`myCar`对象，它将总是有这3个属性。程序清单9.2展示了在声明这些属性并且给它们赋值之后如何访问它们。

---

```
1: <?php
2: class myCar {
3:     public$color = "silver";
4:     public$make = "Mazda";
5:     public$model = "Protege5";
6: }
7: $car = new myCar();
8: echo "I drive a: ".$car -> color." ".$car -> make." ".$car -> model;
9: ?>
```

---

如果把这段代码保存为objproperties.php，将其放置在文档根目录下，使用Web浏览器访问它，将会看到如下的显示结果。

```
I drive a: silver Mazda Protege5
```

由于你总是开着一辆银灰色的Mazda Protege5的机会很小，所以你想要改变myCar对象的属性。程序清单9.3示意了如何做到这一点。

---

```
1: <?php
2: class myCar {
3:     public$color = "silver";
4:     public$make = "Mazda";
5:     public$model = "Protege5";
6: }
7: $car = new myCar();
8: $car -> color = "red";
9: $car -> make = "Porsche";
10: $car -> model = "Boxter";
11: echo "I drive a: ".$car -> color." ".$car -> make." ".$car -> model;
12: ?>
```

---

如果把这段代码保存为objproperties2.php，将其放置在文档根目录下，使用Web浏览器访问它，将会看到如下的显示结果。

```
I drive a: red Porsche Boxter
```

提示：

在这个例子中，即便\$color、\$make和\$model属性在声明的时候没有初始值，第8行到第10行还是会为它们赋一个值。只要属性声明了，随后就可以使用它们，不管有没有初始值。

程序清单9.3的目的是为了展示只要拥有了定义好的具有属性的类，就可以很容易地修改这些属性的值以满足自己的需要。

### 9.1.2 对象方法

方法为对象添加了功能，对象不再只是静静地坐着，只是保存属性而任岁月蹉跎，它们会实际地做一些事情。程序清单9.4展示了这一点。

程序清单9.4 具有方法的一个类

```
1: <?php
2: class myClass {
3:     function sayHello() {
4:         echo "HELLO!";
5:     }
6: }
7: $object1 = new myClass();
8: $object1 -> sayHello();
9: ?>
```

尽管这不是方法的最激动人心的例子，如果把这段代码保存为helloclass.php，将其放置在文档根目录下，使用Web浏览器访问它，将会看到如下的显示结果。

HELLO!

因此，一个方法的外观及其行为就像一个正常的函数，只不过方法是定义在一个类的框架中的。->操作符用来在脚本中调用对象方法。只

要类中定义了变量，方法都能够访问它们以供自己使用。程序清单9.5说明了这一点。

程序清单9.5 在一个方法中访问类属性

---

```
1: <?php
2: class myClass {
3:     public$name = "Jimbo";
4:     function sayHello() {
5:         echo "HELLO! My name is ".$this->name;
6:     }
7: }
8: $object1 = new myClass();
9: $object1 -> sayHello();
10: ?>
```

---

如果把这段代码保存为helloclass.php，将其放置在文档根目录下，使用Web浏览器访问它，将会看到如下的显示结果。

```
HELLO! My name is Jimbo
```

正如我们在第5行看到的，特殊的变量\$this用来引用当前实例化的对象。任何时候，如果一个对象引用自身，都必须使用\$this变量。把\$this变量和->操作符结合起来使用，我们就能够在类自身之中访问其任何属性或方法。

有关对象属性基础用法的最后一个小例子，就是如何在一个方法中更改一个属性。而在前面，属性都是在包含它的方法之外修改。程序清单9.6展示了如何做到这一点。

程序清单9.6 在一个方法中改变一个属性的值

---

```
1: <?php
2: class myClass {
3:     public$name = "Jimbo";
4:     function setName($n) {
5:         $this->name = $n;
6:     }
7:     function sayHello() {
8:         echo "HELLO! My name is ".$this->name;
9:     }
10: }
11: $object1 = new myClass();
12: $object1 -> setName("Julie");
13: $object1 -> sayHello();
14: ?>
```

---

如果把这段代码保存为`helloclass3.php`，将其放置在文档根目录下，使用Web浏览器访问它，将会看到如下的显示结果。

```
HELLO! My name is Julie
```

为什么？因为在第4~6行中，创建了一个名为`setName()`的新函数。当在第12行调用它的时候，它把`$name`的值修改为Julie。因此，当在第13行调用`sayHello()`函数的时候，它查找`$this->name`，它使用了Julie，这是刚刚由`setName()`函数设置的新值。换句话说，一个对象能够修改自己的属性，在这个例子中，就是`$name`变量。

### 9.1.3 构造方法

构造方法（`constructor`）是存在于类中的一个函数，它和类同名，当使用`new classname`创建类的一个新实例的时候，自动调用构造方法。使用构造方法，我们能够为类提供参数，当类调用的时候，参数就会立即处理。我们将在下一节看到关于对象继承的构造方法的应用。



## 9.2 对象继承

学习了对象、属性和方法的基础之后，我们可以开始看看对象继承。和类相关的继承是名副其实的，一个类可以从其父类那里继承功能。程序清单9.7给出了一个例子。

程序清单9.7 继承其父类的一个类

---

```
1: <?php
2: class myClass {
3:     public$name = "Matt";
4:     function myClass($n) {
5:         $this->name = $n;
6:     }
7:     function sayHello() {
8:         echo "HELLO! My name is ".$this->name;
9:     }
10: }
11: class childClass extends myClass {
12: //code goes here
13: }
14: $object1 = new childClass("Baby Matt");
15: $object1 -> sayHello();
16: ?>
```

---

如果把这段代码保存为inheritance.php，将其放置在文档根目录下，使用Web浏览器访问它，将会看到如下的显示结果。

```
HELLO! My name is Baby Matt
```

第4~6行就是一个构造函数。注意，这个函数的名字和包含它的类的名字相同，都是myClass。第11~13行又定义了一个类，即childClass，它没有包含代码。这是不错的，因为在本例中，它的意义只在于展示从父类的继承。继承通过使用extends子句来发生，如第11行所示。由于使用了这个子句，第二个类继承了第一个类的元素。

最后一个例子展示了子类如何覆盖父类的方法，参见程序清单9.8。

程序清单9.8 子类的方法覆盖了父类的方法

---

```
1: <?php
2: class myClass {
3:     public$name = "Matt";
4:     function myClass($n) {
5:         $this->name = $n;
6:     }
7:     function sayHello() {
8:         echo "HELLO! My name is ".$this->name;
9:     }
10: }
11: class childClass extends myClass {
12:     function sayHello() {
13:         echo "I will not tell you my name.";
14:     }
15: }
16: $object1 = new childClass("Baby Matt");
17: $object1 -> sayHello();
18: ?>
```

---

和程序清单9.7中的代码相比，这段代码唯一的改变就在于第12～14行。在这些行中，创建了一个名为sayHello()的函数，显示了消息“I will not tell you my name”，而不是显示“HELLO! My name is...”。现在，由于sayHello()函数存在于childClass中，而childClass是第16行所调用的类，那么，就使用它的sayHello()版本。

如果把这段代码保存为inheritance2.php，将其放置在文档根目录下，使用Web浏览器访问它，将会看到如下的显示结果。

```
I will not tell you my name
```

就像面向对象程序设计的大多数要素一样，当试图让代码变得更灵活的时候，继承是有用的。假设你创建了一个文本格式化类，它组织和

存储数据并在HTML中格式化它们，并且将结果输出到浏览器，于是你拥有了个人的杰作。现在，假设有一个客户想要使用这一想法，但是他需要把内容格式化为纯文本并保存到一个文本文件中，而不是把内容格式化为HTML并发送给浏览器。没问题，只需要添加几个方法和属性，然后你就完事了。最后，那位客户跑来说他其实想要把数据格式化并作为一封E-mail发送。这是怎么搞得，为什么没有创建XML格式化文件呢？

尽管你可能灰心丧气，但其实这并不是什么难事。如果把编译和存储类与格式化类分隔开，对于每一种发布方法（HTML、文本、Email和XML）都是如此，你就基本拥有了一个父类-子类关系。包含编译和存储方法的那个是父类。负责格式化的是子类，它们从父类继承信息并且根据自己的功能来输出结果。于是所有人都满意了。

## 9.3 小结

本章提供了使用面向对象代码的一个基础。这些内容肯定无法涵盖面向对象程序设计的方方面面，大学里已经有教授关于这一主题的一系列课程，因此，你可能觉得这里的篇幅是有点少。然而，你学到了如何创建类以及从类实例化对象。你学习了如何创建和访问一个类的属性和方法，如何构建新的类，以及如何从父类继承功能。这些也并不少。

## 9.4 Q&A

**Q:** 为什么我在属性声明中看到是**var**而不是**public**、**private**或**protected**。

**A:** 在PHP的较早版本中，**var**用来在类中声明属性。为了向后兼容，如果你的代码中仍然使用**var**，会将其当作**public**对待而不会引发任何问题（除非你想要它成为**private**或**protected**属性）。

**Q:** 要成为一名好的**PHP**程序员，或者说只是要读完本书，我必须要理解面向对象程序设计吗？

**A:** 并非如此。实际上，本书中的项目都是过程式的而不包含面向对象的程序设计。面向对象程序设计是以提高组成一个给定应用程序的代码的重用性和扩展性为目的的一种组织方法。在一个面向对象设计的完整规划的最初阶段，你可能对项目知道的还不够多。当它完成后，或者至少达到一个稳定的状态时，你才能够开始看看在哪些领域可以采取面向对象的方法，并且可以开始把代码组织成类、属性和方法。但是，大多数时候，执行特定任务的简单脚本并没有用面向对象的方式去编写，除非你本身就带有面向对象程序设计的背景。

要更多地了解PHP的对象模型，请参见PHP手册的相关部分，在<http://www.php.net/manual/en/language.oop5.php> 可以找到。

## 9.5 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 如何声明一个名为emptyClass的类，它没有方法也没有属性？
2. 应该怎样为构造方法选取名字？
3. 如果一个变量声明为private，那么，它可以在哪里使用。

## 解答

1. 使用class关键字即可。

```
class emptyClass {  
}
```

2. 你没法选择，构造方法使用它所在的类的名字。
3. 声明为private的变量只能在类自身中使用。



## 思考题

1. 创建一个名为baseCalc的类，它存储了两个数字作为属性。接下来，创建一个calculate()方法，该方法向浏览器显示数字。
2. 为思考题1中的类创建名为addCalc()、subCalc()、mulCalc()和divCalc()的方法，它们覆盖了calculate()方法并且向浏览器显示相应的计算结果。

## 第3部分 深入编程

第10章 使用字符串、日期和时间

第11章 使用表单

第12章 使用**Cookie**和用户会话

第13章 使用文件和目录

第14章 使用图像

## 第10章 使用字符串、日期和时间

在本章中，你将学到：

- 如何格式化字符串；
- 如何确定一个字符串的长度；
- 如何在一个字符串中查找子串；
- 如何把一个字符串分解为组成部分；
- 如何从一个字符串的开头和结尾删除空格；
- 如何替换子字符串；
- 如何改变一个字符串的大小写；
- 如何获取当前日期和时间；
- 如果获取有关日期和时间的信息；
- 如何格式化日期和时间信息；
- 如何验证日期的有效性；
- 如何设置日期和时间。

不管Web内容变得多么丰富，其背后都是HTML在把基于字符串的内容显示到Web浏览器。因此，PHP提供了很多函数用来格式化和操作字符串，这也就毫不令人意外了。类似的，日期和时间是每天生活中很重要的一部分，以至于我们不必多想也很容易地用到它们。然而，由于公历（Gregorian calendar）很难直接使用，PHP提供了功能强大的工具使得对日期的操作成为一项容易的任务。

有很多的**PHP**函数可供使用，本章不会介绍所有这些函数。然而，本章还是对如何使用某些基本的字符串、日期和时间函数提供了一个基础，并且针对如何考虑在代码中使用这些函数提供了一个想法。首要的原则是，如果你想要转换、操作或显示一个字符串、日期或时间，在参考**PHP**手册之前，不要尝试去构建自己的解决方案，因为，可能已经存在一个函数可以完成你想要完成的任务。

## 10.1 使用PHP格式化字符串

直到目前，我们接触的都是简单地以其最初状态直接显示到浏览器的字符串。PHP提供了两个函数，允许我们格式化字符串，把双精度数舍入到指定的小数位数，或在一个字段中定义斜体，或根据不同的数值系统来显示数据。在本节中，我们将看到printf()和sprintf()所提供的一些格式化选项。

### 10.1.1 使用printf()

如果你具有类似C语言的程序设计语言的经验，就会熟悉printf()函数的概念。printf()函数需要一个字符串作为参数，叫做格式控制字符串（format control string）。它还接受附加的不同类型的参数，我们将稍后学习。格式控制字符串包含了与显示这些附加参数相关的指令。例如，如下的代码段，使用printf()来把一个整数输出为八进制的（或者以8为基数的）数。

```
<?php
printf("This is my number: %o", 55);
// prints "This is my number: 67"
?>
```

在格式控制字符串（第一个参数）内，我们包含了一个特殊的代码，这就是转换指示（conversion specification）。转换指示以一个百分号(%)打头，并且定义了如何对待printf()相应的参数。我们可以在格式控制字符串中包含任意多个转换指示，只要向printf()发送相等数目的参数就可以了。

如下的代码段使用printf()输出两个浮点数。

```
<?php
printf("First number: %f<br/>Second number: %f<br/>", 55, 66);
// Prints:
// First number: 55.000000
// Second number: 66.000000
?>
```

第一个转换指示对应printf()的第一个附加参数，即55，第二个转换指示对应66。跟在百分号后面的f要求把数据当作一个浮点数，转换指示的这个部分叫做类型指定符（type specifier）。

printf()和类型指定符

我们已经遇到了两种类型指定符，以八进制显示整数的o，以及以浮点数显示整数的f。表10-1列出了其他可用的类型指定符。

表10-1      类型指定符

指 定 符	说 明
d	以十进制数显示参数
b	以二进制数显示一个整数
c	以对等的ASCII显示一个整数
f	以浮点数（双精度）显示一个整数
o	以八进制数（以8为基数）显示一个整数

s	以字符串显示参数
x	以一个小写十六进制数（以16为基数）显示一个整数
X	以一个大写十六进制数（以16为基数）显示一个整数

程序清单10.1使用printf()来根据表10-1列出的某个类型指定符显示一个单个的数字。注意，我们不仅向格式控制字符串添加了转换指示，包含其中的任何附加文本也将显示出来。

程序清单10.1 显示某些类型指定符

---

```
1: <?php
2: $number = 543;
3: printf("Decimal: %d<br/>", $number);
4: printf("Binary: %b<br/>", $number);
5: printf("Double: %f<br/>", $number);
6: printf("Octal: %o<br/>", $number);
7: printf("String: %s<br/>", $number);
8: printf("Hex (lower): %x<br/>", $number);
9: printf("Hex (upper): %X<br/>", $number);
10: ?>
```

---

把上述代码放到一个名为printfest.php的文本文件中，并且把这些文件放到Web服务器文档根目录下。当通过Web浏览器访问这些脚本的时候，会产生如图10-1所示的输出。正如你所看到的，printf()是一种把数据从一种数值系统转换为另一种数值系统并将结果输出的快捷方法。

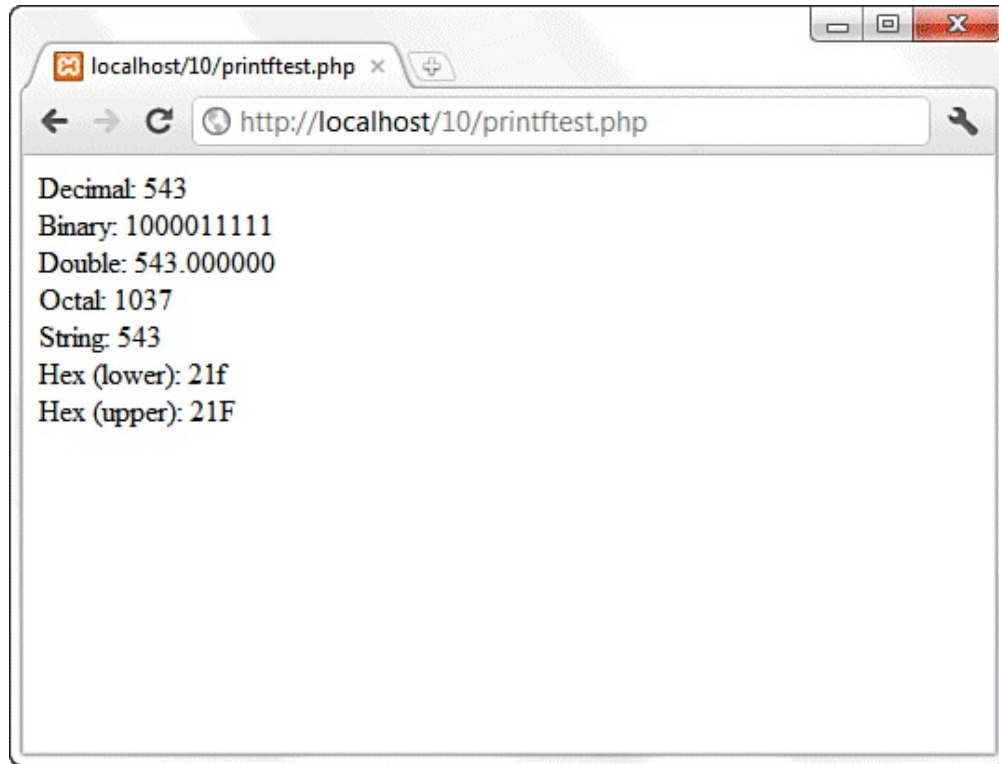


图10-1 显示转换指示

在HTML中指定一种颜色的时候，我们把3个00到FF之间的十六进制数组合起来，它们分别表示红、绿和蓝。我们可以使用printf()把0到255之间的3个十进制数转换为相等的十六进制形式，示例如下。

```
<?php
$red = 204;
$green = 204;
$blue = 204;
printf("#%X%X%X", $red, $green, $blue);
// prints "#CCCCCC"
?>
```

尽管我们可以使用类型指定符来实现从十进制到十六进制的数字转换，但我们无法使用它来确定每个参数在输出中占多少个字符。在一个HTML颜色代码中，每个十六进制数字应该占两个字符，如果我们把前面代码段中的变量\$red、\$green和\$blue的值改为1，那将产生问题，得



到的输出结果是#111。可以通过使用填充指示符（padding specifier）强制输出前面以0打头。

## 使用填充指示符填充输出

我们可能需要通过一个占位符来填充输出。填充指示符应该直接跟在百分号后面，这个百分号是一个转换指示的开头。要使用占位的0来填充输出，填充指示符应该由一个0以及跟在后面的我们希望输出占用的字符个数来组成。如果输出所占用的字符少于这个总数，之前的差额将会用0填充，示例如下。

```
<?php
printf("%04d", 36);
// prints "0036"
?>
```

要使用占位空格来填充输出，填充指示符由一个空格以及跟在后面的表示输出应该占用的字符数目组成，示例如下。

```
<?php
printf("% 4d", 36)
// prints "  36"
?>
```

### 提示：

浏览器不会显示一个HTML文档中的多个空格。可以通过在输出的外围放置

```
标记来强制显示空格和新行，示例如下。
```

```
<pre>
<?php
echo "The      spaces      will be visible";
?>
</pre>
```

如果想要把一个完整的文档格式化为文本，可以使用header()函数来改变Content-Type标头，示例如下。

```
header("Content-Type: text/plain");
```

别忘了，为了让header()函数如期地工作，我们不得向浏览器发送任何输出。可以在填充指示符中指定空格或0以外的任何字符，只需要用一个单引号后面跟着你想用的字符就行了，示例如下。

```
<?php
printf ("%'x4d", 36);
// prints "xx36"
?>
```

我们现在有了完成HTML代码示例所需的工具。在此之前，我们可以把3个数字转换为十六进制数，可以不用0填充，但不能用占位0来填充十六进制值，示例如下。

```
<?php
$red = 1;
$green = 1;
$blue = 1;
printf("#%02X%02X%02X", $red, $green, $blue);
// prints "#010101"
?>
```

每个变量都作为十六进制数输出。如果输出占位少于两位，将会添加0。

### 10.1.2 指定一个字段宽度

我们可以指定输出应该占用的位数。字段宽度指示符（field width specifier）是一个整数，假设没有定义填充指示符，它应该放置在表示转换指示的百分号的后面。如下的代码段输出4个项目的一个列表，所有的这些都在一个20位的字段中。为了让这个位置在浏览器中可见，我们把所有的输出放置到一个pre元素中。

```
<?php
echo "<pre>";
printf("%20s\n", "Books");
printf("%20s\n", "CDs");
printf("%20s\n", "DVDs");
printf("%20s\n", "Games");
printf("%20s\n", "Magazines");
echo "</pre>";
?>
```

图10-2给出了这个代码段的输出。

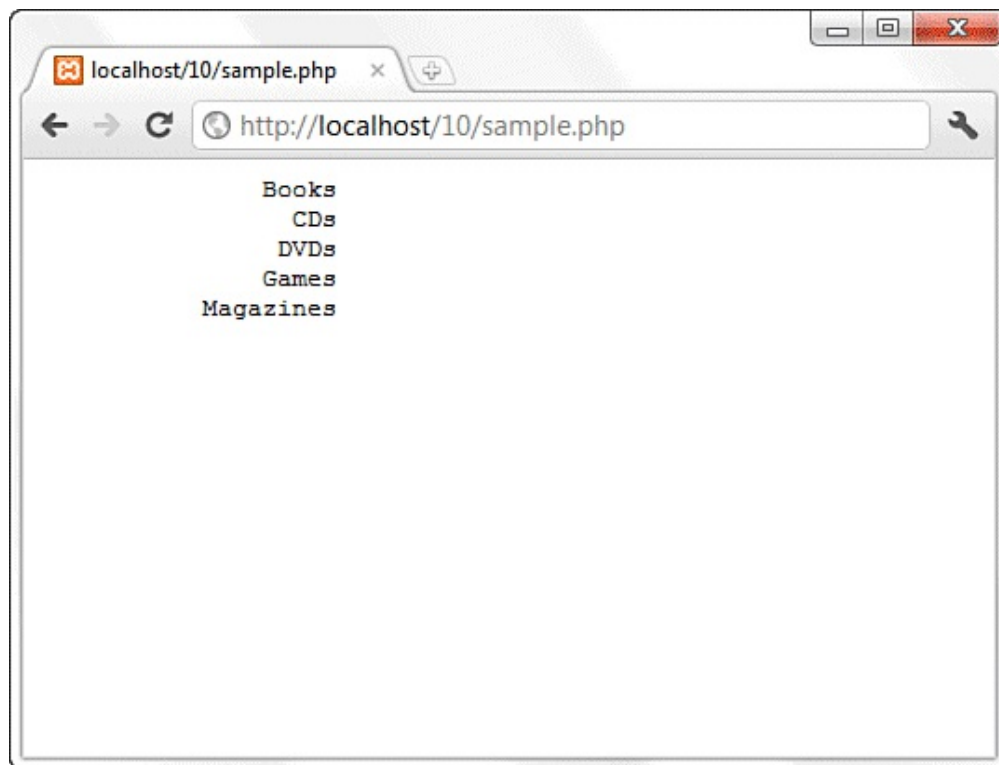


图10-2 使用字段宽度指示符对齐

默认情况下，输出在你所指定的字段中是右对齐的。我们可以通过  
在字段宽度指示符前面加一个减号，从而让其左对齐。

```
printf("%-20s\n", "Left aligned");
```

注意，对齐方式应用到输出中的任何数字的小数部分。换句话说，  
当使用右对齐的时候，只有双精度数的小数点前面的部分和字段宽度的

末尾对齐。

指定精度

如果想要以浮点数格式输出数据，可以指定自己想要对数据舍入的精度。当处理货币的时候，这非常有用。精度指示符应该直接放置在类型指示符的前面。它包含一个点以及后面跟着的想要舍入的小数位数。这个指示符只对带有类型指示符f的输出中的数据有效，示例如下。

```
<?php
printf("%.2f", 5.333333);
// prints "5.33"
?>
```

提示：

在C语言中，可以和printf()一起使用一个精度指示符来指定小数输出的填充。这个精度指示符对于PHP中的小数输出没有作用。使用这个填充指示符将添加整数前的占位0。

转换指示：复习

表10-2列出了可以组成一个转换指示的指示符，按照它们可能出现的顺序。注意，很难同时使用一个填充指示符和一个字段宽度指示符，可以选择使用其中的一个，但不能同时使用。

表10-2      转换指示的组成部分

名      字	说      明	示      例
填充指示符	确定输出应该占用的字符数，以及如果占位不够应该添加的字符	'4'

字段宽度指示符	确定输出格式化后的宽度	'20'
精度指示符	确定了一个双精度数应该舍入的小数位数	‘.4’
类型指示符	确定应该输出的数据类型	'd'

程序清单10.2使用printf()来输出产品价格的一个列表。

程序清单10.2 使用printf()来格式化产品价格的一个列表

```
1:  <?php
2:  $products = array("Green armchair" => "222.4",
3:                    "Candlestick"=> "4",
4:                    "Coffee table"=> "80.6");
5:  echo "<pre>";
6:  printf("%-20s%20s\n", "Name", "Price");
7:  printf("%'-40s\n", "");
8:  foreach ($products as $key=>$val) {
9:      printf( "%-20s%20.2f\n", $key, $val );
10: }
11: echo "</pre>";
12: ?>
```

我们首先在第2行到第4行定义包含产品名和定价的一个关联数组。在第5行显示出pre元素的开始标记，以使浏览器可以识别空格和换行。在第6行对printf()的第一次调用使用了如下的格式控制字符串。

`"%-20s%20s\n"`

格式控制字符串中的第一个转换指示（“%-20s”）定义了一个20个字符的宽度指示符，而且输出是左对齐的。还使用了字符串类型指示符。第二个转换指示（“%20s”）设置了一个右对齐的字段宽度。这个

`printf()`调用将会输出我们的字段标题。

第7行的`printf()`函数调用显示了包含“-”字符的一行，它有40个字符的宽度。我们使用一个填充指示符来实现这一点，它会为空白字符串添加填充。

第9行的`printf()`函数调用，是遍历产品数组的`foreach`语句的一部分。在这里使用两个转换指示，第一个（“%-20s”）把产品名显示为一个字符串，它在20个字符的一个字段内左对齐；而第二个（“%20.2f”）使用一个字段宽度指示符来确保输出右对齐的20个字符的一个字段。我们还使用了一个精度指示符来确保输出舍入到两位小数。

把上述代码放入到一个名为`printftest2.php`的文本文件中，并且把这些文件放入到Web服务器文档根目录下。当通过Web浏览器访问这些脚本的时候，会产生如图10-3所示的输出。



图10-3 使用printf()格式化产品和价格

### 10.1.3 参数交换

假设要把日期输出到浏览器。我们把日期存储在一个多维数组中，并且使用printf()来格式化输出，示例如下。

```
<?php
$dates = array(
    array('mon'=> 12, 'mday'=>25, 'year'=>2011),
    array('mon'=> 1, 'mday'=>23, 'year'=>2012),
    array('mon'=> 10, 'mday'=>29, 'year'=>2011)
);
$format = include("local_format.php");
foreach($dates as $date) {
    printf("$format", $date['mon'], $date['mday'], $date['year']);
}
?>
```

在前面的代码段中，格式控制字符串来自一个名为local\_format.php的包含文件。假设这个文件只包含如下内容。

```
<?php
return "%02d/%02d/%d<br/>";
?>
```

我们的输出将遵守如下格式mm/dd/yyyy。

```
12/25/2011
01/23/2012
10/29/2011
```

假设现在我们要在欧洲的一个站点安装我们的脚本，那里的日期格式通常是日期显示在月份前面（dd/mm/yyyy）。假设核心代码已经写定并且无法修改，我们该怎么办呢？幸运的是，我们现在可以改变参数在格式控制代码中的顺序，只需要把return语句修改如下。

```
return "%2\%02d/%1\%02d/%3\%d<br/>";
```

可以在每个转换指示的百分号后面插入我们所想要的参数数目，后面跟着一个转义的美元符号(\$)。这样，在代码段中，可以要求先显示如下第二个参数，

```
%2\%02d
```

接着是第一个参数，

```
%1\%02d
```

最后是第三个参数。

```
%3\%d
```

新的代码的结果就是如下以英文格式列出日期。



25/12/2011  
23/01/2012  
29/10/2011

### 10.1.4 存储一个格式化字符串

`printf()`函数把数据输出到浏览器，这意味着结果不能供脚本使用。然而，我们可以使用`sprintf()`函数，这个函数的使用方法和`printf()`一样，只不过它返回一个可以存储以供稍后使用的字符串。如下的代码段使用`sprintf()`来把一个双精度数舍入为两位小数，并把结果存储到`$cash`中。

```
<?php
$cash = sprintf("%.2f", 21.334454);
echo "You have \$$cash to spend.";
// Prints "You have $21.33 to spend."
?>
```

我们专门使用`sprintf()`来把格式化的数据写入到一个文件中，你可以调用`sprintf()`并且将其返回值赋予一个变量，可以用`fputs()`把这个变量输出到一个文件中。

## 10.2 了解PHP中的字符串

我们并不总是了解所使用的数据的一切。字符串可以有很多来源，包括用户输入、数据库、文件和Web页面。在开始操作来自外部源的数据之前，常常需要了解有关数据的更多信息。PHP提供了很多函数，我们可以用来获取有关字符串的信息。

### 10.2.1 索引字符串的一个注意事项

我们常常需要使用和字符串相关的一个词，“索引”（index）。在第8章中讨论数组的时候已经遇到过这个词。实际上，字符串和数组并不是想象的那样差异迥然。我们可以把字符串看作是字符的一个数组，因此，我们可以像访问数组的元素那样，访问一个字符串的单个字符，示例如下。

```
<?php
$test = "phpcoder";
echo $test[0]; // prints "p"
echo $test[4]; // prints "o"
?>
```

字符就像是数组的元素，也有一个从0开始而不是从1开始的索引。当我们讨论一个字符串中的字符的位置或索引的时候，记住上面这一点是很重要的。

### 10.2.2 使用strlen()获取一个字符串的长度

我们可以使用内建的strlen()函数来确定一个字符串的长度。这个函数需要一个字符串作为参数，并且返回一个整数，表示我们传递给它的

这个字符串中的字符的数目。`strlen()`可以用来检查用户输入的长度，就像在下面的代码段中，它检测一个会员代号以确保其确实是4位数字的长度。

```
<?php
$membership = "pAB7";
if (strlen($membership) == 4) {
    echo "<p>Thank you!</p>";
} else {
    echo "<p>Your membership number must be four characters long.</p>";
}
?>
```

只有当`$membership`变量所保存的字符串的长度确实为4位的时候，用户才会得到感谢信息。否则，将会显示一条出错信息。

### 10.2.3 使用`strstr()`获取一个字符串的子串

我们可以使用`strstr()`函数来测试一个字符串是否存在于另一个字符串之中。这个函数需要两个参数，源字符串和我们想要在其中查找的子字符串。如果没有找到子字符串，这个函数返回`false`，否则，它返回源字符串的一部分，这部分以子字符串开始。在下面的例子中，假设对于那些会员代码中包含了字符串`AB`的会员和没有包含字符串`AB`的会员，我们要区分对待。

```
<?php
$membership = "pAB7";
if (strstr($membership, "AB")) {
    echo "<p>Your membership expires soon!</p>";
} else {
    echo "<p>Thank you!</p>";
}
?>
```

由于变量`$membership`的值包含了子字符串`AB`，`strstr()`函数返回字

字符串AB7。测试的时候，这个函数得到的结果是true，因此，我们显示相应的信息：“Your membership expires soon!”。但如果我们用来查找的源字符串是“pab7”，那会发生什么情况？由于strstr()是区分大小写的，将不会找到AB。最初的if语句的测试将会失败，浏览器将会显示默认消息(“Thank you!”)。如果我们想要在字符串中查找AB或ab，必须使用substr()来代替strstr()，该函数的用法完全相同，但是它的查找不区分大小写。

### 10.2.4 使用strpos()找到一个子字符串的位置

strpos()函数告诉我们在一个大字符串中是否存在一个字符串，并且告诉我们其位置。strpos()函数需要两个参数：源字符串和要查找的子字符串。这个函数还接受一个可选的第三个参数，这是一个整数类型的值，表示要从哪里开始查找的索引。如果这个子字符串不存在，strpos()的返回值为false，否则，它返回子字符串开始的索引。下面的代码段使用strpos()来确保一个字符串以字符串mz开始。

```
<?php
$membership = "mz00xyz";
if (strpos($membership, "mz") === 0) {
    echo "Hello mz!";
}
?>
```

注意我们用来获得期望的结果的技巧。尽管strpos()函数在我们的字符串中查找mz，它从第一个元素开始查找，也就是0位置。如果strpos()函数的返回值为false，那么在if条件测试中，false也将返回0。为了解决这一点，我们使用了等同运算符===，如果左边操作数和右边的操作数相等并且具有相同的类型，就像在本例中的情况一样，才会返回true。

这只是用来在haystacks中查找needles的一些和字符串相关函数的几个变体之一。参考PHP手册中所提供的关于这个函数的内容，还可以找到其他相关的函数。

### 10.2.5 使用substr()提取一个字符串的一部分

substr()函数根据开始索引和我们要查找的字符串的长度来返回一个字符串。这个函数需要两个参数：源字符串和开始索引。使用这些参数，函数会返回从开始索引到我们所查找的字符串的末尾的所有字符。我们也可以（可选地）提供第三个参数，表示要返回的字符串的长度的整数。如果使用了第三个参数，substr()只是从开始索引向前，返回指定数目的字符。

```
<?php
$test = "phpcoder";
echo substr($test,3)."<br/>"; // prints "coder"
echo substr($test,3,2)."<br/>"; // prints "co"
?>
```

如果传递给substr()一个负数作为第二个参数（开始索引），它将会从字符串的末尾而不是开头开始计算。下面的代码为那些已经提交了一个以.fr结尾的Email地址的人编写了一条特定的消息。

```
<?php
$test = "pierre@wanadoo.fr";
if ($test = substr($test, -3) == ".fr") {
    echo "<p>Bonjour! Nous avons des prix spéciaux de vous.</p>";
} else {
    echo "<p>Welcome to our store.</p>";
}
?>
```

### 10.2.6 使用strtok()分解一个字符串

我们使用strtok()函数一个单词一个单词地解析一个字符串。这个函

数需要两个参数：需要分解的字符串以及用来分隔字符串的分隔符。分隔符字符串可以包含任意多个字符，并且这个函数将返回所找到的第一个分解。在第一次调用`strtok()`函数后，源字符串将放入缓存，对于后续的调用，只需要给`strtok()`函数传递分隔符字符串。每次调用这个函数时，它将会返回所找到的下一个分解，当到达字符串末尾的时候返回`false`。`strtok()`函数通常都在一个循环中重复地调用。程序清单10.3使用`strtok()`来分解一个URL，首先从查询字符串中分解出主机和路径，并进一步分解出查询字符串的名/值对。

程序清单10.3 使用`strtok()`分解一个字符串

---

```
1: <?php
2: $test  = "http://www.google.com/search?";
3: $test .= "hl=en&ie=UTF-8&q=php+development+books&btnG=Google+Search";
4: $delims = "?&";
5: $word = strtok($test, $delims);
6: while (is_string($word)) {
7:     if ($word) {
8:         echo $word."<br/>";
9:     }
10:    $word = strtok($delims);
11: }
12: ?>
```

---

把上述代码放入到一个名为 `teststrtotok.php` 的文本文件中，并且把这些文件放入到Web服务器文档根目录下。当通过Web浏览器访问这些脚本的时候，会产生如图10-4所示的输出。

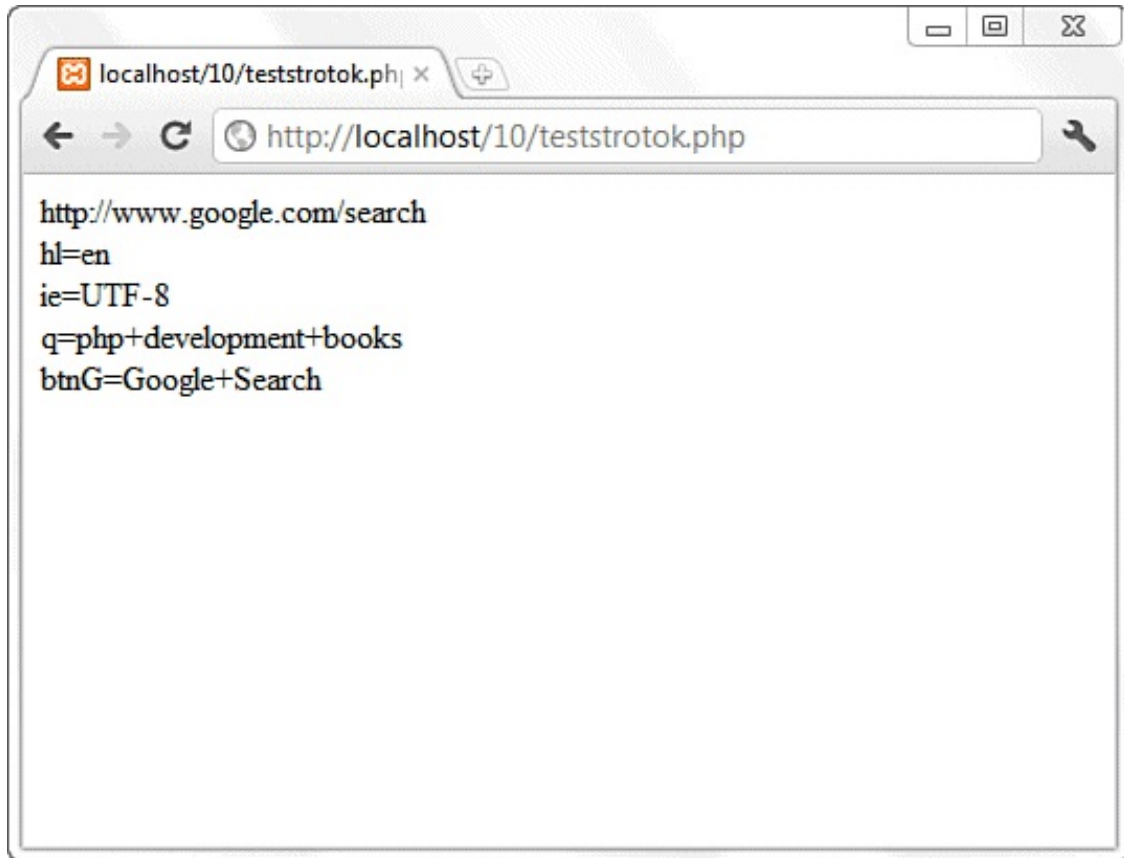


图10-4 teststrotok.php的输出，一个分解后的字符串

`strtok()`函数像一个有点机械的指令，使用它需要一些技巧。在第4行，我们首先把想要使用的分隔符存储到一个变量中，即`$delims`。在第5行调用`strtok()`，把想要分解的URL以及`$delims`字符串作为参数传递给它，我们把第一个结果存储到`$word`中。在第6行的`while`循环的条件测试中，我们测试`$word`是否是一个字符串。如果不是，便可知道已经到达字符串的末尾，并且不再需要进行任何操作。

我们测试返回类型的原因，是因为这样一种情况：一个字符串在一行中包含两个分隔符，当遇到这两个分隔符中的第一个的时候，会导致`strtok()`返回一个空字符串。因此，再次进行下面的测试。

```
while ($word) {  
    $word = strtok($delims);  
}
```

如果\$word是一个空字符串，将会失效，即便还没有到达源字符串的末尾。

既然已经确定了\$word包含一个字符串，我们可以继续使用它。如果\$word没有包含一个空字符串，在第8行将其显示到浏览器。随后，必须在第10行再次调用strtok()为下一次测试重新填充\$word变量。注意，我们不要再次把源字符串传递给strtok()，如果这么做，源字符串的第一个单词将会再次返回，这将会进入一个无限循环。



## 10.3 在PHP中操作字符串

PHP提供了很多函数，可以用来转换一个字符串参数，不管是略作改变还是彻底改变。我们将很快看到它们。

### 10.3.1 使用**trim()**、**ltrim()**和**strip\_tags()**整理一个字符串

当用户输入或者从一个外部文件获取文本的时候，我们总是无法确定是否在数据的开头和结尾提取了空白字符。**trim()**函数从一个字符串的开头和末尾剔除了任何空白字符，包括换行符、Tab符和空格。它接受要修改的字符串作为参数，返回一个整理后的版本，示例如下。

```
<?php
$text = "\t\tlots of room to breathe    ";
echo "<pre>$text</pre>";
// prints "      lots of room to breathe    ";
$text = trim($text);
echo "<pre>$text</pre>";
// prints "lots of room to breathe";
?>
```

当然，这可能不是我们所期望的工作。我们可能希望保留一个字符串开头的空白而删除其末尾的空白。我们可以使用PHP的**rtrim()**函数来完成和使用**trim()**所做的同样的事情。但是，只有字符串末尾的空白被删除掉了，示例如下。

```
<?php
$text = "\t\tlots of room to breathe    ";
echo "<pre>$text</pre>";
// prints "      lots of room to breathe    ";
$text = rtrim($text);
echo "<pre>$text</pre>";
// prints "      lots of room to breathe";
?>
```

PHP提供了ltrim()函数，它只从一个字符串的开头删除空白。同样，它也使用我们所要转换的字符串来调用，并且返回一个新的字符串，其中删除了Tab键、换行和空白，示例如下。

```
<?php
$text = "\t\tlots of room to breathe    ";
echo "<pre>$text</pre>";
// prints "      lots of room to breathe    ";
$text = ltrim($text);
echo "<pre>$text</pre>";
// prints "lots of room to breathe    ";
?>
```

从一段文本中删除标记，从而不带HTML格式地显示字符串，这是很常见的需求。PHP提供了 strip\_tags()函数来达到此目的。strip\_tags()函数接受两个参数：要转换的文本和一个可选的strip\_tags()可以保留的HTML标记的集合。这个列表中的标记不应该用任何字符隔开，示例如下。

```
<?php
$string = "<p>\"I <em>simply</em> will not have it,\" <br/>said Mr Dean.</p>
<p><strong>The end.</strong></p>";
echo strip_tags($string, "<br/><p>");
?>
```

在上面的代码段中，我们创建了一个HTML格式的字符串。当调用 strip\_tags()的时候，就把\$string变量和一个例外的标记列表传递给它。结果是<p> 和<br>标记保留，而所有其他的标记都删除掉。注意，<p>的配套标记</p>也删除了。

这个代码段的输出如下。

```
"I simply will not have it,"
said Mr Dean.
```

```
The end.
```

注意，斜体和粗体的格式没有了，但段落和换行依然保留。

### 10.3.2 使用**substr\_replace()**替换一个字符串的一部分

**substr\_replace()**函数和**substr()**函数的功能类似，只是它允许我们把所提取出的那部分字符串替换掉。这个函数需要3个参数：要修改的字符串、需要添加给它的文本以及开始索引。它还接受一个可选的长度参数。**substr\_replace()**函数在字符串中查找由开始索引和长度参数所指定的部分，用所提供的字符串替换这一部分，然后返回整个修改后的字符串。

下面的代码段用来更新一个用户的会员代码，我们修改了其中第三个和第四个字符。

```
<?php
$membership = "mz11xyz";
$membership = substr_replace($membership, "12", 2, 2);
echo "New membership number: $membership";
// prints "New membership number: mz12xyz"
?>
```

这段代码的结果是：旧的会员号码“mz11xyz”被转换为新的会员号码“mz12xyz”。

### 10.3.3 使用**str\_replace()**替换子字符串

**str\_replace()**函数用来把另一个字符串中的给定字符串的所有实例都替换掉。它需要3个参数：一个查找字符串、替换字符串和主字符串。这个函数返回修改后的字符串。

如下的例子使用**str\_replace()**把主字符串中所有出现2010的地方修改

为2012。

```
<?php
$string = "<h1>The 2010 Guide to All Things Good in the World</h1>";
$string .= "<p>Site contents copyright 2010.</p>";
echo str_replace("2010","2012",$string);
?>
```

str\_replace()函数的所有参数都接受数组和字符串的形式的值，这就允许我们对一个主字符串甚至在多个主字符串上执行多次查找和替换操作。以下面的代码段为例。

```
<?php
$source = array(
    "The package which is at version 4.2 was released in 2005.",
    "The year 2005 was an excellent time for PointyThing 4.2!");
$search = array("4.2", "2005");
$replace = array("6.3", "2012");
$source = str_replace($search, $replace, $source);
foreach($source as $str) {
    echo "$str<br>";
}
?>
```

这段代码的输出如下。

```
The package which is at version 6.3 was released in 2012.
The year 2012 was an excellent time for PointyThing 6.3!
```

当一个字符串数组传递给str\_replace()作为第一个和第二个参数的时候，它会尝试在要修改的文本中，把每个查找字符串和相应的替换字符串作交换。如果第三个参数是一个数组，str\_replace()函数返回一个字符串数组。在返回的字符串数组中的每个字符串上已经执行了查找和替换操作。

### 10.3.4 转换大小写

PHP提供了几个函数，允许我们转换字符串的大小写,改变大小写通常对字符串比较有用。要得到一个字符串的大写版本，使用函数 `strtoupper()`。这个函数只需要一个想要转换的字符串，并且返回转换后的字符串，示例如下。

```
<?php
$membership = "mz11xyz";
$membership = strtoupper($membership);
echo "$membership"; // prints "MZ11XYZ"
?>
```

要把一个字符串转换为小写字符，使用 `strtolower()` 函数。同样，这个函数只需要一个想要转换的字符串，并且返回转换后的字符串，示例如下。

```
<?php
$membership = "MZ11XYZ";
$membership = strtolower($membership);
echo "$membership"; // prints "mz11xyz"
?>
```

PHP还提供了一个具有很好的装饰作用的大小写函数。 `ucwords()` 函数使得字符串中的每个单词的第一个字母变为大写。在下面的例子中，我们将使得用户提交的字符串中的每个单词的第一个字母变为大写。

```
<?php
$full_name = "violet elizabeth bott";
$full_name = ucwords($full_name);
echo $full_name; // prints "Violet Elizabeth Bott"
?>
```

尽管这个函数使得每个单词的第一个字母变为大写，但它并没有动其他的字母。因此，在前面的例子中，如果用户的Shift键有问题，并且提交了 `VIolEt eLIZaBeTH bOTt`， `ucwords()` 函数对于修正字符串不会做太多的工作。我们将最终得到 `VIolEt ELIZaBeTHBOTt`，这谈不上有太

大的改善。我们可以在调用ucwords()之前通过strtolower()函数把提交的字符串转换为小写，从而解决这个问题，示例如下。

```
<?php
$full_name = "Violet eLIzBeTH bOTt";
$full_name = ucwords(strtolower($full_name));
echo $full_name; // prints "Violet Elizabeth Bott"
?>
```

最后，ucfirst()函数只是把一个字符串的第一个字母变为大写。在下面的代码段中，我们把一个用户提交的字符串的首字母大写，示例如下。

```
<?php
$myString = "this is my string.";
$myString = ucfirst($myString);
echo $myString; // prints "This is my string."
?>
```

使用大小写相关的字符串函数是有用的，例如在试图验证区分大小写的密码时。如果用户输入“MyPass”，而存储的密码是“mypass”，但我们希望这个匹配是不区分大小写的，可以尝试将用户输入的小写或大写版本与存储的密码的小写或大写版本进行比较。如果他们在同样大小写格式下是匹配的，用户可以通过验证，即使他输入的内容和实际存储的内容有一些不同。

### 10.3.5 使用wordwrap()和nl2br()换行文本

当在一个Web页面中显示纯文本的时候，常常会面临一个问题，换行没有显示并且文本乱成一团。nl2br()函数方便地把每个换行符转换为一个HTML分段，示例如下。

```
<?php
$string = "one line\n";
$string .= "another line\n";
$string .= "a third for luck\n";
echo nl2br($string);
?>
```

上述代码输出结果如下。

```
one line<br />
another line<br />
a third for luck<br />
```

nl2br()函数只是对于保持要转换的文本中已经存在的换行符效果很好。有时候，我们可能需要添加任意的换行符来格式化一串文本，wordwrap()函数是完成这一任务的最好选择。wordwrap()需要一个参数，即要转换的字符串。默认情况下，wordwrap()函数把每75个字符分为一行，并且使用\n作为换行符。因此，示例代码段如下。

```
<?php
$string = "Given a long line, wordwrap() is useful as a means of ";
$string .= "breaking it into a column and thereby making it easier to read";
echo wordwrap($string);
?>
```

上述代码段将输出如下的结果。

```
Given a long line, wordwrap() is useful as a means of breaking it into a
column and thereby making it easier to read
```

由于这些行用字符\n断开，故格式化并不会以HTML代码形式显示。当然，wordwrap()函数还有两个可选参数：表示每行最大字符数的一个数字以及要用来表示一行结束的字符串。因此，如下的函数调用

```
echo wordwrap($string, 24, "<br/>\n");
```

应用到我们的\$string变量，输出结果如下。

```
Given a long line,<br/>
wordwrap() is useful as<br/>
a means of breaking it<br/>
into a column and<br/>
thereby making it easier<br/>
to read
```

如果有一个单词的字符超出了限制，wordwrap()函数并不会在行限制的地方自动断行。然而，我们可以使用第四个可选的参数来强制实现这一点，这个参数应该是一个正整数。因此，使用带有四个参数的wordwrap()函数，我们就可以换行一个字符串，即便是它所包含的单词超过了我们设置的限制。示例代码段如下。

```
<?php
$string = "As usual you will find me at http://www.witteringonaboutit.com/";
$string .= "chat/eating_green_cheese/forum.php. Hope to see you there!";
echo wordwrap($string, 24, "<br/>\n", 1);
?>
```

上述代码段将输出如下结果。

```
As usual you will find<br/>
me at<br/>
http://www.witteringonab<br/>
outit.com/chat/eating_gr<br/>
een_cheese/forum.php.<br/>
Hope to see you there!
```

上述代码段不会输出如下结果。

```
As usual you will find<br/>
me at<br/>
http://www.witteringonaboutit.com/chat/eating_green_cheese/forum.php. <br/>
Hope to see you there!
```

### 10.3.6 使用explode()把字符串分解到数组

explode()函数和strtok()函数的某些方式类似。但是explode()函数把一个字符串分解到一个数组中，然后，我们可以按照自己的意愿去存



储、排序或查看。`explode()`函数需要两个参数，用来分隔源字符串的分隔字符串以及源字符串本身。这个函数可选地接受第三个参数，这个参数决定了这个字符串最多可以分解为多少部分。分隔字符串可以包含多个字符，所有这些字符构成一个单个的分隔符（不像是传递给`strtok()`的多个分隔字符，其中每个分隔字符自成一体）。下面的代码段分隔一个日期并且把结果存储到一个数组中。

```
<?php
$start_date = "2012-02-19";
$date_array = explode("-", $start_date);
// $date_array[0] == "2012"
// $date_array[1] == "02"
// $date_array[2] == "19"
echo $date_array[0]."-".$date_array[1]."-".$date_array[2];
//prints 2012-02-19
?>
```

现在，我们的脑海里已经充满了一些常见的PHP字符串函数，接下来将介绍日期和时间函数。

## 10.4 使用PHP中的日期和时间函数

下面的小节介绍PHP中与日期和时间相关的函数。自己尝试列表中的每个函数，可以发现这些函数是如此简单而且功能强大。

### 10.4.1 使用time()获取日期

PHP的time()函数提供了所有需要知道的关于当前日期和时间的信息。它不需要参数，并且返回一个整数。对于我们人类而言，返回的数字有点难以识别，但无论如何，它特别有用。

```
echo time();  
// sample output: 1326853185  
// this represents January 17, 2012 at 09:19PM
```

time()返回的整数表示从GMT 1970年1月1日午夜开始流逝过的秒数。这个时刻叫做UNIX时间戳（UNIX Epoch），从那时起已流逝的秒数就叫做时间戳（time stamp）。PHP提供了优秀的工具来把一个时间戳转换为人类所习惯的形式。即便如此，你可能还是会问，“一个时间戳难道不是存储日期的一种无用而费解的方式吗？”实际上恰恰相反。仅仅从一个数字，我们就可以提取出很多的信息。更何况时间戳使得日期的计算比我们想象的更加容易。

设想一个自制的日期系统，其中我们记录了一个月的第几天以及月份和年份。现在，假设一个脚本必须为一个给定的日期添加一天，如果这个日期正好是1999年12月31日，我们必须编写代码把当月的天数设置为1，把月份设置为1月，把年份设置为2000，而不是在原来的日期上加1。使用一个时间戳，我们只需要在当前的数值上添加一天所包含的秒

数（60×60×24，或86400），这就完成了任务。在我们需要的时候，可以把这个新数字转换为某种更友好的形式。

### 10.4.2 使用getdate()转换一个时间戳

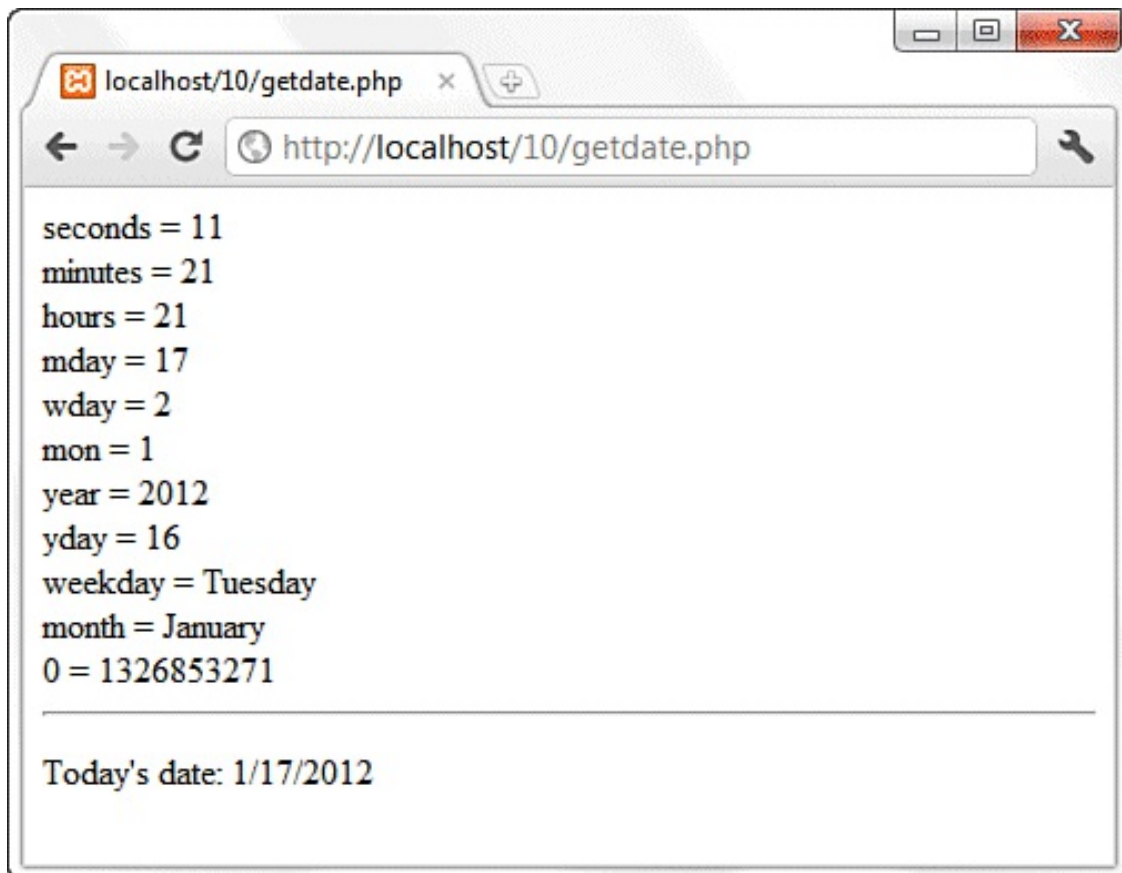
现在我们可以使用一个时间戳了，在将其展示给用户之前，我们必须转换它。getdate()函数可选地接受一个时间戳，并且返回包含了有关日期信息的一个关联数组。如果省略了时间戳，getdate()使用time()所返回的当前时间戳。表10-3列出了包含在getdate()所返回的数组中的元素。

表10-3 getdate()所返回的关联数组

键	说 明	示 例
seconds	一分钟里已经过去的秒数（0～59）	53
minutes	一小时中已经过去的分钟数（0～59）	44
hours	一天中的小时数（0～23）	17
mday	一月中的天数（1～31）	3
wday	一周中的天数（0～6）	0
mon	一年中的月数（1～12）	2
year	年份（4位数字）	2008

yday	一年中的天数（0~365）	33
weekday	星期几（名字）	Sunday
month	一年中的月份（名字）	February
0	时间戳	1092065443

程序清单10.4在第2行使用`getdate()`来从一个时间戳中提取信息，在第3行用一条`foreach`语句来显示每个元素。我们可以看到图10-5所示的典型输出。`getdate()`函数根据本地时区返回日期。



```
seconds = 11
minutes = 21
hours = 21
mday = 17
wday = 2
mon = 1
year = 2012
yday = 16
weekday = Tuesday
month = January
0 = 1326853271

Today's date: 1/17/2012
```

图10-5 使用getdate()

程序清单10.4 使用getdate()获取日期信息

---

```
1: <?php
2: $date_array = getdate(); // no argument passed so today's date will be used
3: foreach ($date_array as $key => $val) {
4:     echo "$key = $val<br>";
5: }
6: ?>
7: <hr/>
8: <?php
9: echo "<p>Today's date: ".$date_array['mon']. "/" . $date_array['mday']. "/" .
10:     $date_array['year']. "</p>";
11: ?>
```

---

如果运行这段代码，将会导致如下所示的一条警告。

```
Warning: getdate(): It is not safe to rely on the system's timezone settings.
You are *required* to use the date.timezone setting or the
date_default_timezone_set() function.
```

你需要对php.ini文件做一些修改。特别是，找到如下的内容的位置。

```
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone =
```

将date.timezone的值修改为你的时区（参见<http://php.net/date.timezone> 以获取有效值的一个列表），示例如下。

```
; Defines the default timezone used by the date functions
; http://php.net/date.timezone
date.timezone = America/New_York
```

确保在对php.ini做出修改后重新启动Apache。

### 10.4.3 使用date()转换一个时间戳

当我们想要使用函数输出的日期元素的时候，可以使用getdate()函数。但是，有时候却想要把日期显示为一个字符串，date()函数可实现返回一个表示日期的、格式化的字符串。通过必须传递给date()的字符串参数，我们可以实现对其返回值的众多的格式控制。除了格式化字符串，date()还可选地接受一个时间戳。表10-4列出了一个格式化字符串可能包含的一些代码。我们可以在<http://www.php.net/date> 找到完整的列表。包含在传递给date()的格式字符串中的任何其他数据都包含在了返回值中。

表10-4      使用date()的某些格式化代码

格 式	说 明	示 例
a	am或pm（小写）	am
A	AM或PM（大写）	AM
d	一月中的第几天（用0填充空白的数字）	28
D	星期几（3个字母）	Tue
E	时区标识符	America/Los_Angeles
F	月份名	February
h	小时（12小时格式—前面的空白用0填充）	06

H	小时（24小时格式—前面的空白用0填充）	06
G	小时（12小时格式—前面的空白不用0填充）	6
G	小时（24小时格式—前面的空白不用0填充）	6
i	分钟	45
j	一月中的第几天（用0填充前面的空白的数字）	28
l	星期几（名字）	Tuesday
L	闰年（1表示是，0表示不是）	0
m	一年中的月份（数字——用0填充前面的空白）	02
M	一年中的月份（3个字母）	Feb
N	一年中的月份（数字——不用0填充前面的空白）	2
s	一小时中的秒数	26
S	一月中的天数后面的次序后缀	th
r	符合RFC 822标准的完整日期 （ <a href="http://www.faqs.org/rfcs/rfc822.html">http://www.faqs.org/rfcs/rfc822.html</a> ）	Tue, 28 Feb 2006 06:45:26 -0800

U	时间戳	1141137926
y	年份（两位数）	06
Y	年份（四位数）	2006
z	一年中的天数(0~365)	28
Z	从GMT开始的秒数偏移量	-28800

程序清单10.5试验了上述格式代码中的一部分。

程序清单10.5 使用date()格式化一个日期

---

```

1: <?php
2: $time = time(); //stores the exact timestamp to use in this script
3: echo date("m/d/y G:i:s e", $time);
4: echo "<br/>";
5: echo "Today is ";
6: echo date("jS \of F Y, \a\\t g:ia \i\\n e", $time);
7: ?>

```

---

程序清单10.5两次调用date()：第一次在第3行，输出了一个缩略的日期格式；第二次在第6行调用，产生一个更长一些的格式。把这个程序清单保存到一个名为datetest.php的文本文件中并且在Web浏览器中打开它。日期显然会和下面的有所不同，但是，这里有一些示例输出如下。

```

01/17/12 21:29:31 America/New_York
Today is 17th of January 2012, at 9:29pm in America/New_York

```



尽管这个格式化字符串看上去很神秘，但它很容易构建。如果想要添加一个字符串，其中包含的字母也是格式化代码，我们可以通过在其前面放置一个斜杠来转义它们。对于转义时变成了控制字符的那些字符，必须转义它们前面的反斜杠。例如，`\t`是一个制表符的格式化代码，因此，为了确保显示出制表符，在程序清单10.5所示的例子中使用`\\t`。

另一个例子是，在一些环境下，一个单词要添加到一个字符串中，例如the。单词the是由3个格式化代码组成的，因此必须转义，示例如下。

```
<?php
echo date('l \\t\\h\\e jS');
//prints Tuesday the 3rd
?>
```

还需要注意，`date()`函数根据本地时区返回信息。如果想要用GMT格式化一个日期，我们使用`gmdate()`函数，它工作的方式几乎与`date()`函数相同。

#### 10.4.4 使用**mktime()**创建时间戳

我们已经获得了有关当前时间的信息，但是还无法使用任意的日期。`mktime()`返回一个时间戳，随后我们可以用`date()`或`getdate()`使用它。`mktime()`按照如下的顺序接受6个整型参数。

- 小时
- 分钟
- 秒
- 月份

- 日期
- 年份

程序清单10.6使用mktime()来获取一个时间戳，然后，我们通过date()函数来使用这个时间戳。

程序清单10.6 使用mktime()创建一个时间戳

---

```
1: <?php
2: // make a timestamp for Jan 17 2012 at 9:34 pm
3: $ts = mktime(21, 34, 0, 1, 17, 2012);
4: echo date("m/d/y G:i:s e", $ts);
5: echo "<br/>";
6: echo "The date is ";
7: echo date("jS \of F Y, \a\\t g:ia \i\\n e", $ts );
8: ?>
```

---

我们在第3行调用mktime()并且把返回的时间戳赋给了\$ts变量。然后，在第4行和第7行分别使用date()函数，输出了使用\$ts的日期的格式化版本。我们可以选择省略掉mktime()的一些参数或所有参数，并且用和当前时间对应的值来替代。mktime()还可调整那些超出相应范围的值，因此，一个25的小时参数转换为年份、月份、日期参数指定的日期之后的一天的1:00 am。

把上述代码保存到名为mktimetest.php的文件中并且在Web浏览器中打开它。我们将会看到如下结果。

```
01/17/12 21:34:00 America/New_York
The date is 17th of January 2012, at 9:34pm in America/New_York
```

### 10.4.5 使用checkdate()测试日期

我们可能需要从用户输入接受日期信息。在使用用户输入的日期或

者将其存储到数据库之前，确保这个日期是有效的。`checkdate()`接受3个整型参数：月份、日期和年份。如果月份在1到12之间，并且对于给定的月份和年份（考虑到闰年），这个日期是可以接受的，以及年份在0到32767之间，`checkdate()`返回true。注意，日期可能在`checkdate()`函数中是有效的，但是在其他的日期函数中是不可接受的。例如，如下的代码返回true。

```
checkdate(4, 4, 1066)
```

如果试图使用上述参数值通过`mktime()`来构建一个日期，最终将得到一个-1的时间戳。首要的原则是不要对`mktime()`使用1902年以前的年份，并且，配合日期函数使用1970年以前的任何日期的时候都要小心，因为负数不是有效的日期。由于UNIX时间戳是从1970年1月1日开始的，此前的任何日期都是无效的（负的）时间戳。

## 10.5 其他字符串、日期和时间函数

PHP不会缺乏函数，尤其是对于字符串和日期这样的常用数据类型。查阅PHP手册的以下章节，你会受益匪浅。

- Strings, <http://www.php.net/manual/en/ref.strings.php>
- Date/Time, <http://www.php.net/manual/en/ref.datetime.php>

除了留意添加到PHP中的函数，用户为某些函数撰写的注意事项，往往会为各种编程任务提供解决方案，在我们构建自己的应用程序时会发现这些很有用。

## 10.6 小结

在本章中，我们已经了解到了可以在PHP脚本中控制字符串的一些函数。学习了如何使用printf()和sprintf()格式化字符串，我们应该能够使用这些函数来创建要转换数据的或者格式化数据的字符串。我们学习了获取字符串信息的函数，可以使用strlen()来获取一个字符串的长度，使用 strpos()来判断一个子字符串的存在，或者使用 substr()来提取一个子字符串，还可以使用strtok()来分解一个字符串。

我们还了解了转换字符串的函数。现在可以使用trim()、ltrim()或 rtrim()从字符串的开头或结尾删除空白。可以使用strtoupper()、 strtolower()或ucwords()改变一个字符串中的字符的大小写。可以用 str\_replace()来替换一个字符串中的所有实例。

我们还学习了如何使用各种PHP函数来执行与日期和时间相关的操作。time()函数从当前日期和时间获取一个时间戳，并且，我们可以使用getdate()来从一个时间戳提取日期信息，使用date()把一个时间戳转换为一个格式化字符串。我们学习了如何使用mktime()创建一个时间戳，以及如何使用checkdate()验证日期的有效性。我们还将第16章学习多个功能更强大的、和日期相关的函数，以至于可能发现对于自己的很多与时间日期相关的需求可以使用MySQL而不是PHP。

## 10.7 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 10.8 Q&A

**Q:** 我们可以组合多个字符串函数吗？

**A:** 可以。我们可以嵌套任何函数，而不只是字符串函数。只要记得使用开始括号和结束括号以确保正确地嵌套函数。

**Q:** 对于保存在数据库中而不是脚本中的日期和时间，我该如何使用呢？

**A:** 在本书第16章，我们将学习MySQL日期和时间函数，但是，通常的一种好办法是，如果你要从数据库中提取日期和时间，并且想要执行和日期和时间相关的操作（例如，计算时间，或者将时间从一种类型转换为另一种类型），可以让数据库为你做这些事情。但是，这也并不妨碍你对于从数据库提取的日期来使用PHP的日期和时间函数。

## 问答题

1. 为了使用`printf()`把一个整数格式化为一个双精度小数，应该使用什么转换指示符？给出转换整数33所需的完整语法。
2. 如何使用0填充问题1中的转换，以使小数点前面的部分有4个字符的长度？
3. 如何为上一个问题中已经格式化的浮点数指定一个两位小数的精度？
4. 从一个字符串中提取一个子字符串应该使用什么函数？
5. 如何从一个字符串的开头删除空白？
6. 如何把一个带有分隔符的字符串分解成一个子字符串的数组？
7. 使用PHP时我们怎样获取一个表示当前日期和时间的UNIX时间戳？
8. 哪个PHP函数接受一个时间戳，并且返回一个表示给定日期的关联数组？
9. 使用哪个PHP函数来格式化日期信息？
10. 使用哪个PHP函数来检查一个日期的有效性？



## 解答

1. 用来把一个整数格式化为一个双精度数的转换指示符是f，转换整数33的语法如下。

```
printf("%f", 33);
```

2. 我们可以使用填充指示符来填充printf()输出，即一个空格或一个0后面跟着一个表示我们希望填充的字符数目的数字。

```
printf("%04f", 33);
```

3. 精度指示符由一个点(.)后面跟着一个表示要使用的精度的数字。它应该放在转换指示符的前面。

```
printf("%04.2f", 33);
```

4. substr()函数用来提取并返回一个子字符串。
5. ltrim()函数从一个字符串的开头删除空白。
6. explode()函数把一个字符串分解成一个数组。
7. 使用time()函数。
8. getdate()函数返回一个关联数组，其元素包含了给定日期的各个部分。
9. 使用date()。
10. 使用checkdate()函数来检查一个日期的有效性。

## 思考题

1. 创建一个接受用户的全名和E-mail地址的反馈表单。使用大小写转换函数，把用户提交的每个名字的首字母大写，并且把结果显示到浏览器上。检查用户的E-mail地址是否包含一个@符号，如果不是则显示一条警告。
2. 创建双精度数和整数的一个数组。遍历这个数组，把每个元素都转换为具有2个精度的一个浮点数，结果输出在一个20字符的字段中，并且右对齐。
3. 创建一个生日倒计时脚本。给定一个表单输入月份、日期和年份，输出信息告诉用户距离生日有多少天、多少小时、多少分和多少秒。

## 第11章 使用表单

在本章中，你将学到：

- 如何从表单字段获取信息。
- 如何使用允许多个选择的表单元素。
- 如何创建一个单个的文档，包含一个**HTML**表单以及处理其提交的**PHP**代码。
- 如何保存隐藏字段的状态。
- 如何把用户重定向到一个新的页面。
- 如何构建发送邮件的**HTML**表单和**PHP**代码。
- 如何构建一个上传文件的**HTML**表单，并且编写处理上传的代码。

到现在为止，本书中的**PHP**示例都缺少一个关键要素。当然，我们已经学习了基础知识，可以设置变量和数组，创建并调用函数，并且可以使用字符串。但是，如果用户无法以一种有意义的方式和**Web**站点交互的话，这一切都毫无意义。**HTML**表单就是将大量信息从用户传递给服务器的一种主要手段。因此，在本章中，我们把目光转向这一要素，并且来看看获取和使用用户输入的策略。

## 11.1 创建一个简单的输入表单

从现在开始，让我们把HTML和PHP代码分开。程序清单11.1构建了一个简单的HTML表单。

程序清单11.1 简单的HTML表单

---

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>A simple HTML form</title>
5: </head>
6: <body>
7: <form method="post" action="send_simpleform.php">
8: <p><label for="user">Name:</label><br/>
9: <input type="text" id="user" name="user"></p>
10: <p><label for="message">Message:</label><br/>
11: <textarea id="message" name="message" rows="5" cols="40"></textarea></p>
12: <button type="submit" name="submit" value="send">Send Message</button>
13: </form>
14: </body>
15: </html>
```

---

把上述代码放入到一个名为 `simpleform.html` 的文本文件中，并且将其放置在 Web 服务器文档根目录下。这个代码清单定义了一个表单，它包含在第9行定义的一个名为“user”的文本字段，在第11行定义的一个名为“message”的文本域，以及在第12行定义的一个提交按钮。FORM元素的ACTION参数指向一个名为 `send_simpleform.php` 的文件，它会处理表单信息。这个表单的方法是POST，因此，表单中的数据变量存储在超全局变量 `$_POST` 中。

程序清单11.2创建了接受用户输入的代码。

程序清单11.2 从一个表单读取输入

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>A simple response</title>
5: </head>
6: <body>
7: <p>Welcome, <strong><?php echo $_POST['user']; ?></strong>!</p>
8: <p>Your message is:
9: <strong><?php echo $_POST['message']; ?></strong></p>
10: </body>
11: </html>
```

把上述代码放入到一个名为send\_simpleform.php的文本文件中，并且将其放置在Web服务器文档根目录下。现在使用浏览器访问这个表单本身(simpleform.html)，然后，我们会看到如图11-1所示的结果。



图11-1 simpleform.html创建的表单

当用户提交程序清单11.1所创建的表单时，将会调用程序清单11.2中的脚本。在程序清单11.2的代码中，我们访问了两个变量：\$\_POST

[`'user'`]和`$_POST ['message']`。这些都是对超全局变量`$_POST`中的变量的引用，这些变量包含了用户输入到`user`文本字段和`message`文本域中的值。**PHP**中的表单就是如此之简单。

在表单字段中输入某些信息并单击发送按钮。我们应该看到自己的输入显示在屏幕上。

**提示：**

我们也可以在这个表单以及其他表单中使用`GET`方法。`POST`可以处理比`GET`更多的数据，并且不会把数据传递到查询字符串中。如果你使用了`GET`方法，确保将超全局变量修改为`$_GET`而不是`$_POST`。

## 11.2 使用用户定义数组访问表单输入

前面的例子展示了如何从HTML元素收集信息，这些元素向每个元素名提交一个单独的值，例如文本字段、文本域和单选按钮。当使用像SELECT这样的元素的时候，这就给我们带来一个问题，因为用户可能从一个多选SELECT列表选取一个或多个项目。如果我们使用一个一般的名字来命名SELECT元素，如下所示。

```
<input type="checkbox" id="products" name="products">
```

接受这个数据的脚本只是获取了这个名字(\$\_POST ['products'])对应的一个单个值。我们可以通过重新命名这类元素，使其名字以一对空的方括号结尾，从而改变这种行为。我们在程序清单11.3中这么做。

程序清单11.3 包含一个SELECT元素的HTML表单

---

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>An HTML form with checkboxes</title>
5: </head>
6: <body>
7: <form action="send_formwithcb.php" method="POST">
8: <p><label>Name:</label><br/>
9: <input type="text" name="user" /></p>
10: <fieldset>
11: <legend>Select Some Products:</legend><br/>
12: <input type="checkbox" id="tricorder"
13:     name="products[]" value="Tricorder">
14:     <label for="tricorder">Tricorder</label><br/>
15:
16: <input type="checkbox" id="ORAC_AI"
17:     name="products[]" value="ORAC AI">
18:     <label for="ORAC_AI">ORAC AI</label><br/>
19:
20: <input type="checkbox" id="HAL_2000"
21:     name="products[]" value="HAL 2000">
22:     <label for="HAL_2000">HAL 2000</label>
23: </fieldset>
24: <button type="submit" name="submit" value="submit">Submit Form</button>
25: </form>
26: </body>
27: </html>
```

---

把上述代码放入到一个名为formwithselect.html的文本文件中，并且将其放置在Web服务器文档根目录下。接下来，在处理表单输入的脚本中，我们发现，名为“products []”的所有选中的复选框的值，在一个名为\$\_POST ['products']的数组中是可用的。在第12行到第22行，指定了每个复选框。在程序清单11.4中，可用的用户选择在一个数组中给出。

程序清单11.4 从程序清单11.3的表单中读取输入

---

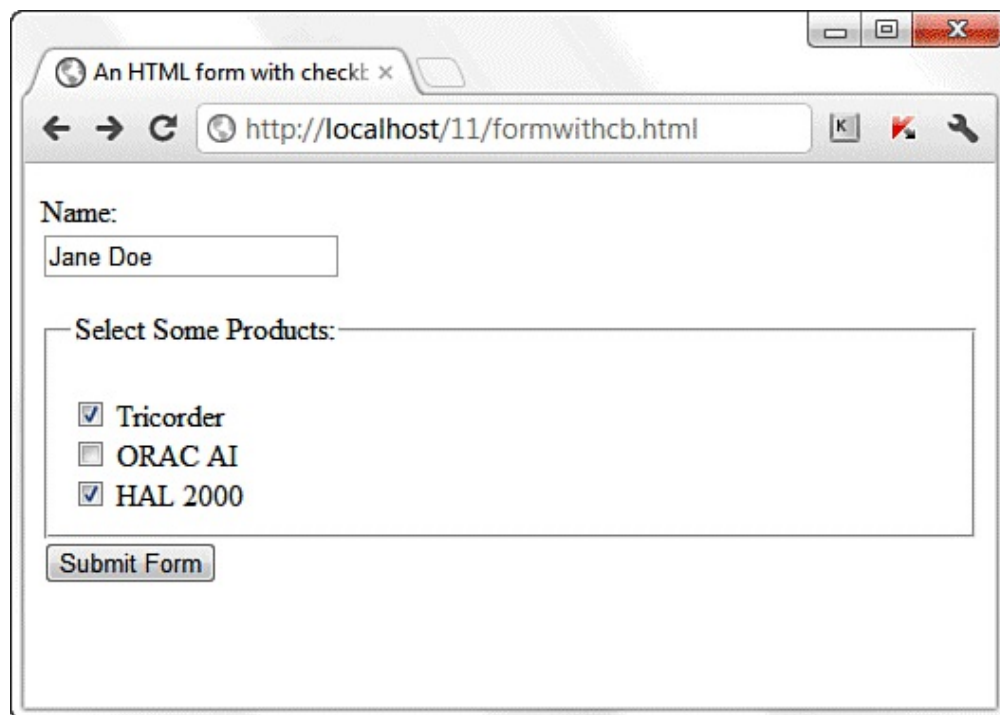
```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>Reading checkboxes</title>
```



```
5: </head>
6: <body>
7: <p>Welcome, <strong><?php echo $_POST['user']; ?></strong>!</p>
8: <p>Your product choices are:
9: <?php
10: if (!empty($_POST['products'])) {
11:     echo "<ul>";
12:     foreach ($_POST['products'] as $value) {
13:         echo "<li>$value</li>";
14:     }
15:     echo "</ul>";
16: } else {
17:     echo "None";
18: }
19: ?>
20: </body>
21: </html>
```

---

把上述代码放入到一个名为end\_formwithselect.php的文本文件中，并且将其放置在Web服务器文档根目录下。现在，通过Web浏览器访问这个表单并且填写字段。图11-2给出一个例子。



An HTML form with check: x

http://localhost/11/formwithcb.html

Name:

Jane Doe

Select Some Products:

☒ Tricorder

☐ ORAC AI

☒ HAL 2000

Submit Form

图11-2 程序清单11.3所创建的表单

在程序清单11.4中的脚本的第7行，我们访问了`$_POST ['user']`变量，这个变量源自user表单元素。在第10行，我们测试了`$_POST ['products']`变量。如果`$_POST ['products']`变量存在，我们在第12行遍历它，并且在第13行把每种选择输出到浏览器。选定的checkbox元素的value属性中的文本成为了存储在数组中的值。

提交表单，结果如图11-3所示。

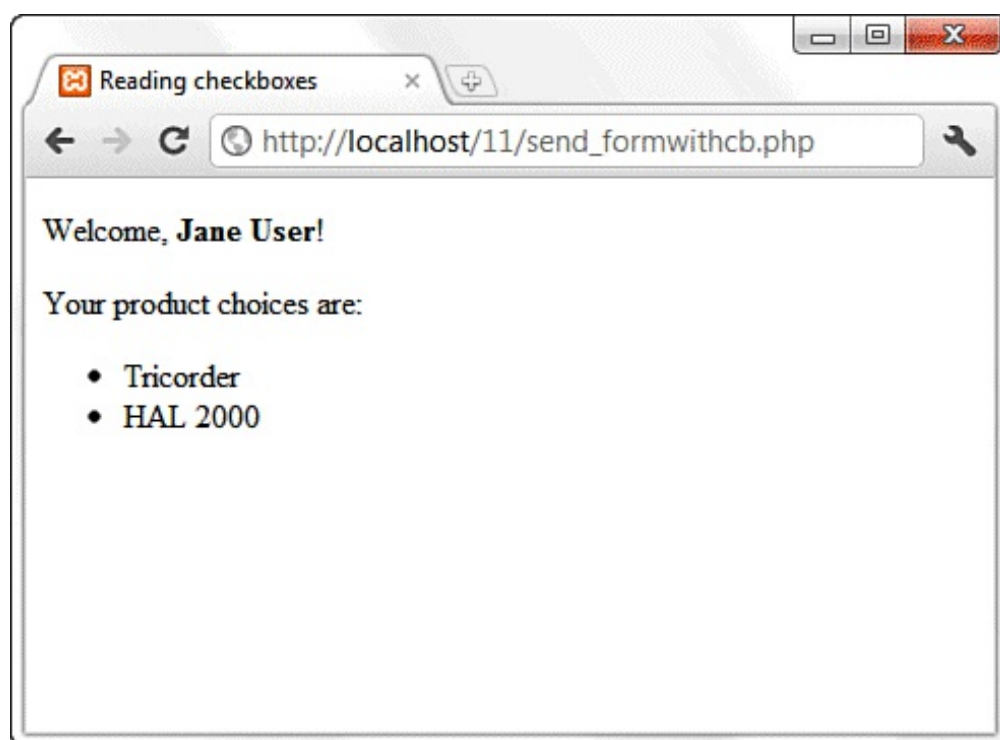


图11-3 send\_formwithselect.php的输出示例

循环技术不仅对于SELECT元素特别有用，也可以对其他类型的表单元元素起作用。例如，通过给定具有相同名字的多个复选框，我们使得一个用户可以在一个单独的字段名中选择多个值。

只要我们选择的名称是以一个空白方括号结束的，PHP都会把这个

字段的用户输入编译到一个数组中。

## 11.3 在单个页面上组合HTML和PHP代码

在某些条件下，我们可能想要把解析表单的PHP代码和直接编码的HTML表单包含到同一个页面中。如果需要对多个用户显示同一个表单的话，这种组合可能会有用。当然，如果动态地编写整个页面，将会有更多的灵活性，但是我们会错过PHP最强大的功能之一，这就是和标准HTML的良好结合。可以在页面中包含的标准HTML越多，设计者或页面开发者修复起来就越轻松，而不必寻找程序员帮忙。

对于下面的例子，假设我们要创建一个站点，它向学龄前儿童教授基本的数学知识。它要求我们创建一个脚本，该脚本从表单输入一个数字，并且告诉用户它比一个预定义的整数大还是小。

程序清单11.5创建了HTML。对于这个例子，我们只需要一个文本字段，但是即便如此，我们还是包含了一些PHP。

程序清单11.5 调用自身的一个HTML表单

---

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>An HTML form that calls itself</title>
5: </head>
6: <body>
7: <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
8: <p><label for="guess">Type your guess here:</label> <br/>
9: <input type="text" id="guess" name="guess" /></p>
10: <button type="submit" name="submit" value="submit">Submit</button>
11: </form>
12: </body>
13: </html>
```

---

正如第7行所示，这个脚本的动作是`$_SERVER['PHP_SELF']`，这

个全局变量显示了当前脚本的名字。换句话说，这个操作告诉脚本重新载入自己。程序清单11.5中的脚本并不产生任何输出，但是，如果向Web服务器上传了这个脚本，访问该页面并察看页面的源代码，我们将会注意到，表单动作现在包含了脚本自身的名字。在程序清单11.6中，我们开始构建这个页面的PHP元素。

程序清单11.6 一个PHP猜数字脚本

---

```
1: <?php
2: $num_to_guess = 42;
3: if (!isset($_POST['guess'])) {
4:     $message = "Welcome to the guessing machine!";
5: } elseif (!is_numeric($_POST['guess'])) { // is not numeric
6:     $message = "I don't understand that response.";
7: } elseif ($_POST['guess'] == $num_to_guess) { // matches!
8:     $message = "Well done!";
9: } elseif ($_POST['guess'] > $num_to_guess) {
10:    $message = $_POST['guess']." is too big! Try a smaller number.";
11: } elseif ($_POST['guess'] < $num_to_guess) {
12:    $message = $_POST['guess']." is too small! Try a larger number.";
13: } else { // some other condition
14:    $message = "I am terribly confused.";
15: }
16: ?>
```

---

首先，必须定义让用户猜的数字，我们在第2行做了这件事情，把42赋给了\$num\_to\_guess变量。接下来，必须定义表单是否提交了，我们可以通过查看变量\$\_POST ['guess']的存在来测试提交，只有在脚本已经用guess字段的一个值提交的时候，这个变量才可用。如果\$\_POST ['guess']的值不存在，我们可以安全地假设访问页面的用户没有提交一个表单。如果这个值存在，我们可以测试该变量所包含的值。对\$\_POST ['guess']变量存在性的测试在第3行进行。

第3行到第15行使用了一个if...else if...else控制结构。根据表单提交了什么内容（如果有提交内容的话），在任何时候，这些条件中只有一

个为true。根据条件，不同的值会赋给\$message变量。随后，该变量在脚本的第18行显示到屏幕上，那是这段脚本的HTML的一部分。

程序清单11.7 一个PHP猜数字脚本（续）

```
17: <!DOCTYPE html>
18: <html>
19: <head>
20: <title>A PHP number guessing script</title>
21: </head>
22: <body>
23: <h1><?php echo $message; ?></h1>
24: <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
25: <p><label for="guess">Type your guess here:</label><br/>
26: <input type="text" is="guess" name="guess" /></p>
27: <button type="submit" name="submit" value="submit">Submit</button>
28: </form>
29: </body>
30: </html>
```

把程序清单11.6和11.7中的所有代码放入到一个名为numguess.php的文本文件中，并且将这些文件放入到Web服务器文档根目录下。现在，使用浏览器访问这些脚本，将看到如图11-4所示的结果。

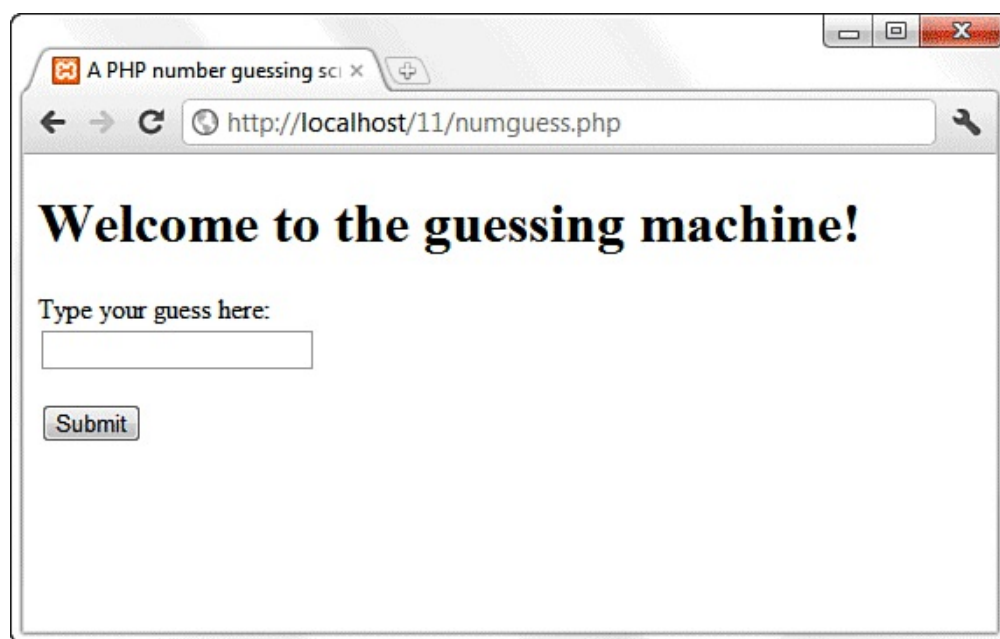


图11-4 程序清单11.6所生成的表单

仍然还有一些额外的事情可以做，但是，你可能会明白，把代码交给一个设计师让他进行艺术加工，这是一件多么简单的事情。设计师可以做好他的那部分工作，而不必以任何方式干扰程序设计，**PHP**代码在最上面，剩下的99%都是**HTML**。

## 11.4 使用隐藏字段来保存状态

通过程序清单11.6中的脚本我们没有办法知道一个用户的猜测是多少，但是，我们可以使用一个隐藏字段来记录这个值。隐藏字段的行为和一个文本字段一样，只不过用户无法看到它，除非查看包含它的文档的HTML源代码。

取出最初的numguess.php脚本并将其保存为一个名为numguess2.php的副本。在新的版本中，在最初给\$num\_to\_guess变量赋值的后面添加如下一行。

```
$num_tries = (isset($_POST['num_tries'])) ? $num_tries + 1 : 1;
```

这一行初始化一个名为\$num\_tries的变量并且为它赋一个值。如果\$\_POST['num\_tries']为空且这个表单还没有提交，\$num\_tries变量的值为1，因为我们将要开始第一次猜数。如果这个表单已经发送，新的值就是\$\_POST['num\_tries']的值加1。

下一个改变在HTML的H1层标题之后。

```
<p><strong>Guess number:</strong> <?php echo $num_tries; ?></p>
```

这个新行只是在屏幕上显示\$num\_tries的当前值。

最后，在表单提交按钮的HTML代码之前，添加隐藏字段。这个字段保存了\$num\_tries递增后的值，如下所示。

```
<input type="hidden" name="num_tries" value="<?php echo $num_tries; ?>" />
```

程序清单11.8显示了完整的新脚本。



---

```
1: <?php
2: $num_to_guess = 42;
3: $num_tries = (isset($_POST['num_tries'])) ? $num_tries + 1 : 1;
4: if (!isset($_POST['guess'])) {
5:     $message = "Welcome to the guessing machine!";
6: } elseif (!is_numeric($_POST['guess'])) { // is not numeric
7:     $message = "I don't understand that response.";
8: } elseif ($_POST['guess'] == $num_to_guess) { // matches!
9:     $message = "Well done!";
10: } elseif ($_POST['guess'] > $num_to_guess) {
11:     $message = $_POST['guess'] . " is too big! Try a smaller number.";
12: } elseif ($_POST['guess'] < $num_to_guess) {
13:     $message = $_POST['guess'] . " is too small! Try a larger number.";
14: } else { // some other condition
15:     $message = "I am terribly confused.";
16: }
17: ?>
18: <!DOCTYPE html>

19: <html>
20: <head>
21: <title>A PHP number guessing script</title>
22: </head>
23: <body>
24: <h1><?php echo $message; ?></h1>
25: <p><strong>Guess number:</strong> <?php echo $num_tries; ?></p>
26: <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
27: <p><label for="guess">Type your guess here:</label><br/>
28: <input type="text" id="guess" name="guess" /></p>
29: <input type="hidden" name="num_tries" value="<?php echo $num_tries; ?>" />
30: <button type="submit" name="submit" value="submit">Submit</button>
31: </form>
32: </body>
33: </html>
```

---

将上述代码保存为numguess2.php文件并且将其放置到Web服务器文档的根目录下。使用Web浏览器访问表单几次，并且尝试猜这个数字（假装你还并不知道它）。每次访问表单，计数器将会增加1。

## 11.5 重定向用户

我们的简单的脚本仍然有一个缺点，不管用户猜测的是否正确，表单都重新载入。**HTML**是直接编码的，这一事实使得很难避免编写整个页面。然而，我们可以把用户重定向到一个祝贺页面，从而避免这个问题。

当一个服务器脚本和一个客户机通讯的时候，它必须首先发送某些标头，这些标头提供了有关后续文档的信息。**PHP**通常会为你自动处理这些标头，但是，可以选择使用**PHP**的**header()**函数发送自己的标头行。

为了调用**header()**函数，我们必须绝对确保没有向浏览器发送输出。第一次向浏览器发送内容的时候，**PHP**送出自己的标头，发送任何更多内容都必须在此之后。任何来自文档的输出，即便是脚本标记外的一个换行或者空格，都会引起标头发送。如果你想要在脚本中使用**header()**函数，必须确保包含函数调用的**PHP**代码之前没有内容。我们还应该检查可能使用的任何其他库。

程序清单11.9展示**PHP**发送给浏览器的一个典型的标头，从第3行开始，作为对第1行中的请求的响应。

程序清单11.9 从一个**PHP**脚本发送的典型的服务器标头

---

```
1: HTTP/1.1 200 OK
2: Date: Sun, 29 Jan 2012 15:50:28 PST
3: Server: Apache/2.2.21 (Win32) PHP/5.4.0
4: X-Powered-By: PHP/5.4.0
5: Connection: close
6: Content-Type: text/html
```

---

通过发送一个Location标头而不是PHP的默认标头，我们可以使浏览器重定向到一个新的页面，示例如下。

```
header("Location: http://www.sampublishing.com");
```

假设我们已经创建了一个名为congrats.html的祝贺页面，我们可以修改猜数字脚本，使得如果猜对了就重定向用户的请求，如程序清单11.10所示。和程序清单11.8相比较，这个脚本只是在第8行else子句之后有些变化。

程序清单11.10 使用header()重定向用户

---

```
1: <?php
2: $num_to_guess = 42;
3: $num_tries = (isset($_POST['num_tries'])) ? $num_tries + 1 : 1;
4: if (!isset($_POST['guess'])) {
5:     $message = "Welcome to the guessing machine!";
6: } elseif (!is_numeric($_POST['guess'])) { // is not numeric
7:     $message = "I don't understand that response.";
8: } elseif ($_POST['guess'] == $num_to_guess) { // matches!
9:     header("Location: congrats.html");
10:    exit;
11: } elseif ($_POST['guess'] > $num_to_guess) {
12:     $message = $_POST['guess'] . " is too big! Try a smaller number.";
13: } elseif ($_POST['guess'] < $num_to_guess) {
14:     $message = $_POST['guess'] . " is too small! Try a larger number.";
15: } else { // some other condition
16:     $message = "I am terribly confused.";
17: }
18: ?>
19:
20: <!DOCTYPE html>
21: <html>
22: <head>
23: <title>A PHP number guessing script</title>
24: </head>
25: <body>
26: <h1><?php echo $message; ?></h1>
27: <p><strong>Guess number:</strong> <?php echo $num_tries; ?></p>
28: <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="POST">
29: <p><label for="guess">Type your guess here:</label><br/>
30: <input type="text" id="guess" name="guess" /></p>
31: <input type="hidden" name="num_tries" value="<?php echo $num_tries; ?>" />
32: <button type="submit" name="submit" value="submit">Submit</button>
33: </form>
34: </body>
35: </html>
```

---

在第8行，我们的if语句的else子句使浏览器把我们送到名为 congrats.html的页面。我们确保当前页面的所有输出都用第10行的exit语句取消，它会立即结束这个脚本的执行和输出。

## 11.6 根据表单提交发送邮件

我们已经看到了如何获取表单响应并且把结果显示到屏幕，距离在一个E-mail消息中发送响应只有一步之遥。然而，在了解如何发送邮件之前，阅读下面的小节，确保已正确地配置了我们的系统。

### 11.6.1 mail()函数的系统配置

在我们可以使用mail()函数发送邮件之前，必须在php.ini文件中设置一些指令，以使该函数能够正常地工作。使用一个文本编辑器打开php.ini，并查找下面这些行。

```
[mail function]
; For Win32 only.
; http://php.net/smtp
SMTP = localhost
; http://php.net/smtp-port
smtp_port = 25

; For Win32 only.
; http://php.net/sendmail-from
;sendmail_from = me@example.com

; For Unix only. You may supply arguments as well (default: "sendmail -t -i").
; http://php.net/sendmail-path
;sendmail_path =
```

如果我们使用Windows作为Web服务器平台，前两条指令适用于我们。为了让mail()函数能够发送邮件，它必须可以访问一个有效的发送邮件服务器。如果你计划使用ISP的发送邮件服务器（在下面的例子中，我们将使用EarthLink），php.ini中的条目应该如下所示。

```
SMTP = smtp.yourisp.net
```

第二条配置指令是`sendmail_from`，这是在发送邮件的From标头中的E-mail地址。它可以在邮件脚本本身中被覆盖，这里通常作为默认值使用，如下面的例子所示。

```
sendmail_from = youraddress@yourdomain.com
```

对于Windows用户来说，一个较好的首要原则是，不管你的Email客户端在机器上安装了什么发送邮件服务器，都应该作为`php.ini`中的SMTP值使用。

如果Web服务器运行在一个Linux/UNIX平台上，可以使用特定机器的`sendmail`功能。在这个例子中，只有最后的命令适合于你，即`sendmail_path`。默认的值是`sendmail -t-i`，但是，如果`sendmail`是临时的或者如果你需要指定不同的参数，也可以不这么做，就像在下面的例子中，它没有使用真实值。

```
sendmail_path = /opt/sendmail -odd -arguments
```

在任何平台上，对`php.ini`做出改变以后，必须重新启动Web服务器以使改变生效。

### 11.6.2 创建表单

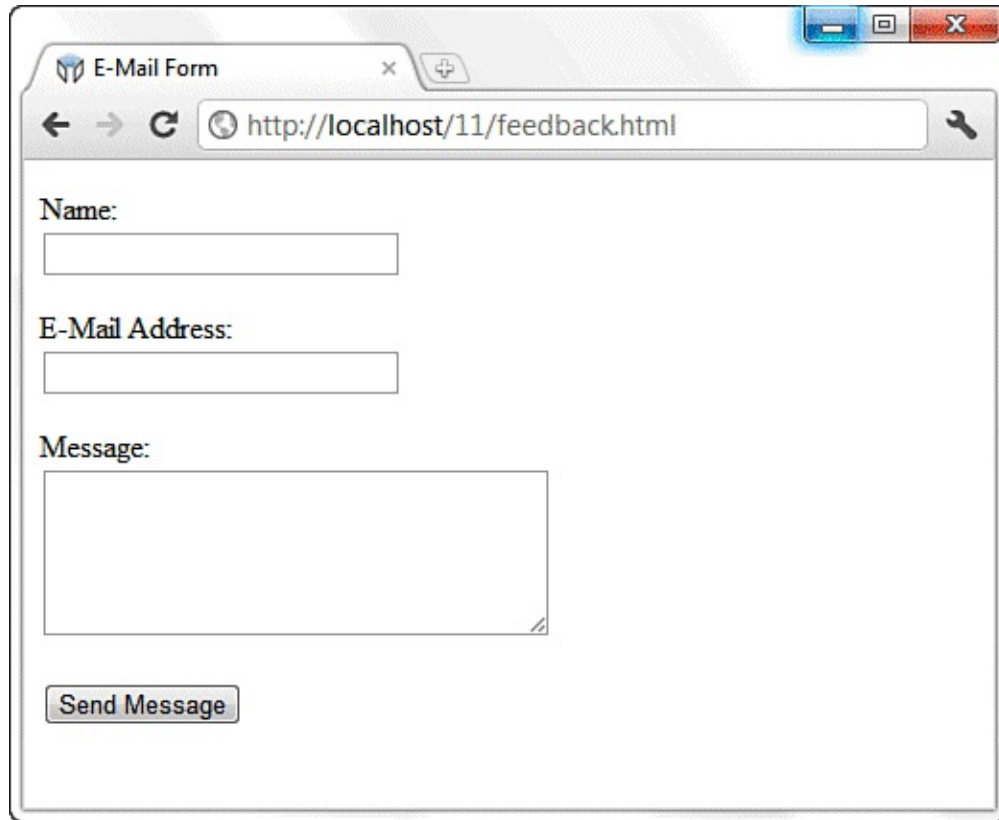
在程序清单11.11中，我们看到用来创建一个简单反馈表单的基本HTML，让我们称其为`feedback.html`。这个表单具有一个`sendmail.php`的action，我们将在下一节中创建它。`feedback.html`中的字段很简单，第8行和第9行创建了一个name字段和标签，第10行和第11行创建了回信的Email地址字段和标签，而第12行和第13行包含了用于用户消息的文本域和标签。

---

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>E-Mail Form</title>
5: </head>
6: <body>
7: <form action="sendmail.php" method="POST">
8: <p><label for="name">Name:</label><br/>
9: <input type="text" size="25" id="name" name="name"/></p>
10: <p><label for="email">E-Mail Address:</label><br/>
11: <input type="text" size="25" id="email" name="email"/></p>
12: <p><label for="msg">Message:</label><br/>
13: <textarea id="msg" name="msg" cols="30" rows="5"></textarea></p>
14: <button type="submit" name="submit" value="send">Send Message</button>
15: </form>
16: </body>
17: </html>
```

---

把上述代码放入到一个名为feedback.html的文本文件中，并且将其放置在Web服务器文档根目录下。当通过Web浏览器来访问这个脚本的时候，它产生如图11-5 所示的结果。



The image shows a web browser window with the title 'E-Mail Form'. The address bar displays 'http://localhost/11/feedback.html'. The form contains three input fields: 'Name:', 'E-Mail Address:', and 'Message:'. Below the fields is a 'Send Message' button.

图11-5 程序清单11.11创建的表单

在下一节中，我们将创建把这个表单发送给接受者的脚本。

### 11.6.3 创建发送邮件的脚本

这个脚本和程序清单11.4中的脚本在概念上只是略有不同，后者只是在屏幕上显示表单响应。在程序清单11.12的脚本中，除了把响应显示到屏幕上，我们还将其发送给一个E-mail地址。

程序清单11.12 发送简单的反馈表单



---

```
1: <?php
2: //start building the mail string
3: $msg = "Name:      ".$_POST['name']."\n";
4: $msg .= "E-Mail:   ".$_POST['email']."\n";
5: $msg .= "Message:  ".$_POST['message']."\n";
6:
7: //set up the mail
8: $recipient = "you@yourdomain.com";
9: $subject = "Form Submission Results";
10: $mailheaders = "From: My Web Site <defaultaddress@yourdomain.com> \n";
11: $mailheaders .= "Reply-To: ".$_POST['email'];
12:
13: //send the mail
14: mail($recipient, $subject, $msg, $mailheaders);
15: ?>
16: <!DOCTYPE html>
17: <html>
18: <head>
19: <title>Sending mail from the form in Listing 11.10</title>
20: </head>
21: <body>
22: <p>Thanks, <strong><?php echo $_POST['name']; ?></strong>,
23:   for your message.</p>
24: <p>Your e-mail address:
25:   <strong><?php echo $_POST['email']; ?></strong></p>
26: <p>Your message: <br/> <?php echo $_POST['message']; ?> </p>
27: </body>
28: </html>
```

---

在第22行到第26行所使用的变量是\$\_POST ['name']、\$\_POST ['email']和\$\_POST ['message'], 它们是表单中字段的名字, 它们的值作为超全局变量\$\_POST的一部分保存。这对于把信息显示到屏幕上来说都是很不错的, 但是在这个脚本中, 我们还需要创建一个在Email中发送的字符串。为此, 我们基本上需要通过连接字符串来形成一条长长的消息字符串从而构成邮件, 并且在适当的地方使用换行符(\n)。

第3行到第5行创建了\$msg变量, 这是包含用户在表单字段中输入的值以及一些额外的说明文字的一个字符串。这个字符串将会形成E-mail的主体。注意在第4行和第5行向\$msg变量添加内容时连接操作符(=)的使用。

第8行和第9行对邮件接受者和邮件消息的主题的变量直接赋值。显然，要用自己的邮件地址来替代`you@yourdomain.com`。如果想要改变邮件的主题，也可以直接去做。

第10行和第11行设置了一些邮件标头，即**From:**和**Reply-to:**标头。你可以把任何值放入到**From:**标头中，这就是当你接受这封邮件的时候显示在发件人或收件人栏的信息。

**提示：**

如果发送邮件服务器是一台Windows机器，那么换行符`\n`应该替换为`\r\n`。

`mail()`函数需要5个参数：收件人、主题、消息、任何附加的邮件标头，以及任何附加的发送邮件参数。在我们的例子中，我们只使用前4个参数。这些参数的顺序如第14行所示。

把上述代码放入到一个名为`sendmail.php`的文本文件中，并且将其放置在Web服务器文档根目录下。使用Web浏览器来访问这个表单，并且输入一些信息，然后单击提交按钮。我们将会在浏览器中看到如图11-6所示的结果。

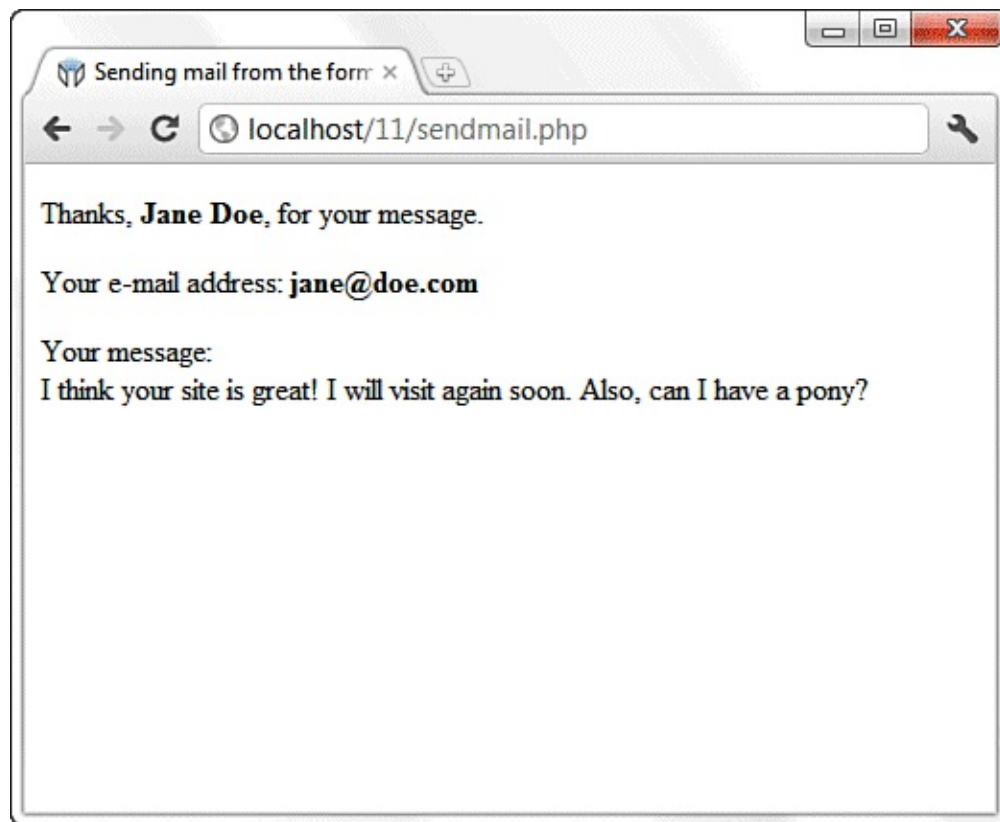


图11-6 来自 sendmail.php的示例结果

如果我们查看自己的E-mail，应该会有一条消息在等着我们去阅读。它看上去如图11-7所示。

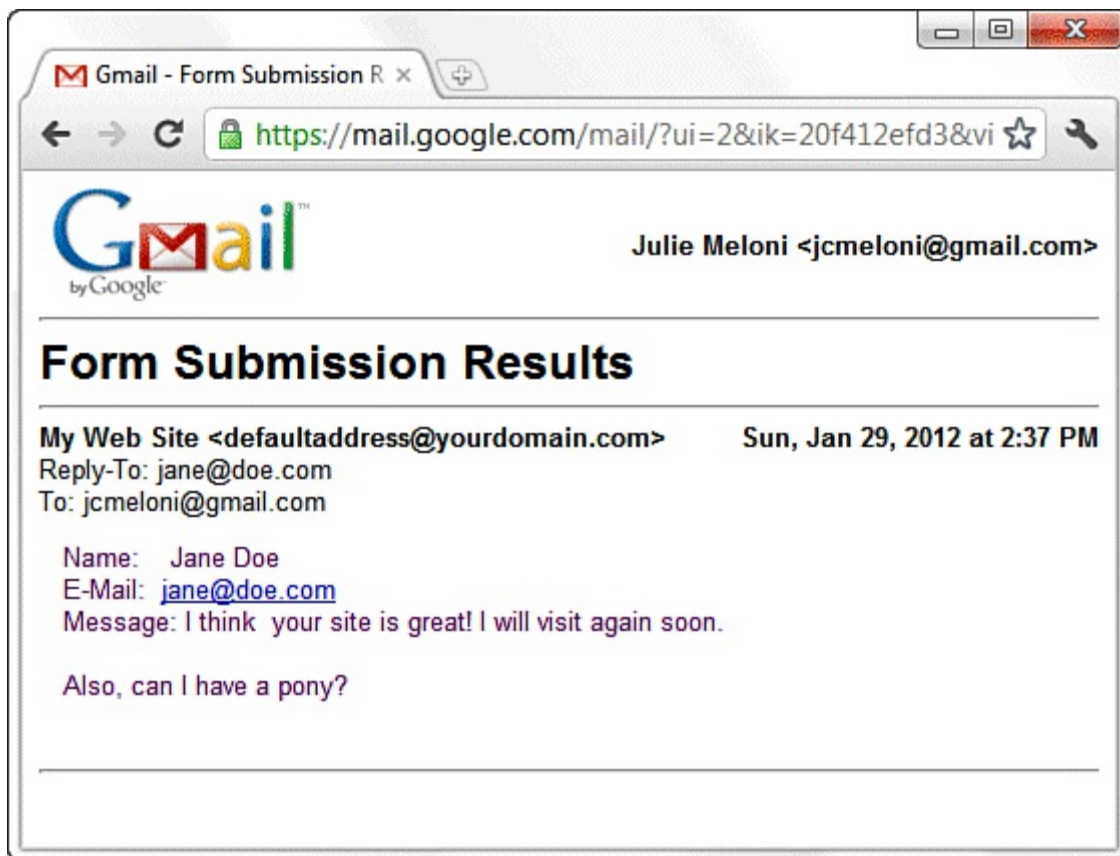


图11-7 sendmail.php发送的邮件

这个示例并不包含任何服务器端的表单元素验证，并且只是假设用户已经在表单中输入了值。在现实的情况中，在对邮件做任何事情之前，你可能要检查表单中是否输入了值以及值的有效性，这需要通过JavaScript或HTML5表单验证。

#### 11.6.4 使用HTML格式化邮件

发送HTML格式的邮件的技巧根本就算不上技巧。实际上，它只是涉及编写实际的HTML和修改mail()函数所发送的标头。作为程序清单11.12的一个改版，程序清单11.13对其第12到14行和第18到19行做出了修改。

---

```
1:  <?php
2:  //start building the mail string
3:  $msg = "<p><strong>Name:</strong>      ".$_POST['name']. "</p>";
4:  $msg .= "<p><strong>E-Mail:</strong>    ".$_POST['email']. "</p>";
5:  $msg .= "<p><strong>Message:</strong>  ".$_POST['message']. "</p>";
6:
7:  //set up the mail
8:  $recipient = "you@yourdomain.com";
9:  $subject = "Form Submission Results";
10: $mailheaders = "MIME-Version: 1.0\r\n";
11: $mailheaders .= "Content-type: text/html; charset=ISO-8859-1\r\n";
12: $mailheaders = "From: My Web Site <defaultaddress@yourdomain.com> \n";
13: $mailheaders .= "Reply-To: ".$_POST['email'];
14:
15: //send the mail
16: mail($recipient, $subject, $msg, $mailheaders);
17: ?>
18: <!DOCTYPE html>
19: <html>
20: <head>
21: <title>Sending the Simple Feedback Form - HTML Version</title>
22: </head>
23: <body>
24: <p>Thanks, <strong><?php echo $_POST['name']; ?></strong>,
25:   for your message.</p>
26: <p>Your e-mail address:
27:   <strong><?php echo $_POST['email']; ?></strong></p>
28: <p>Your message: <br/> <?php echo $_POST['message']; ?> </p>
29: </body>
30: </html>
```

---

在第3行到第5行，消息字符串现在包含了HTML代码。附加的标头在第10行到第11行创建，它将MIME Version标头设置为1.0并将Content-type标头设置为字符集ISO-8859-1的text/html。当在一个支持HTML的邮件客户端上打开它的时候，消息字符串中的HTML将如期地显示，如图11-8所示。

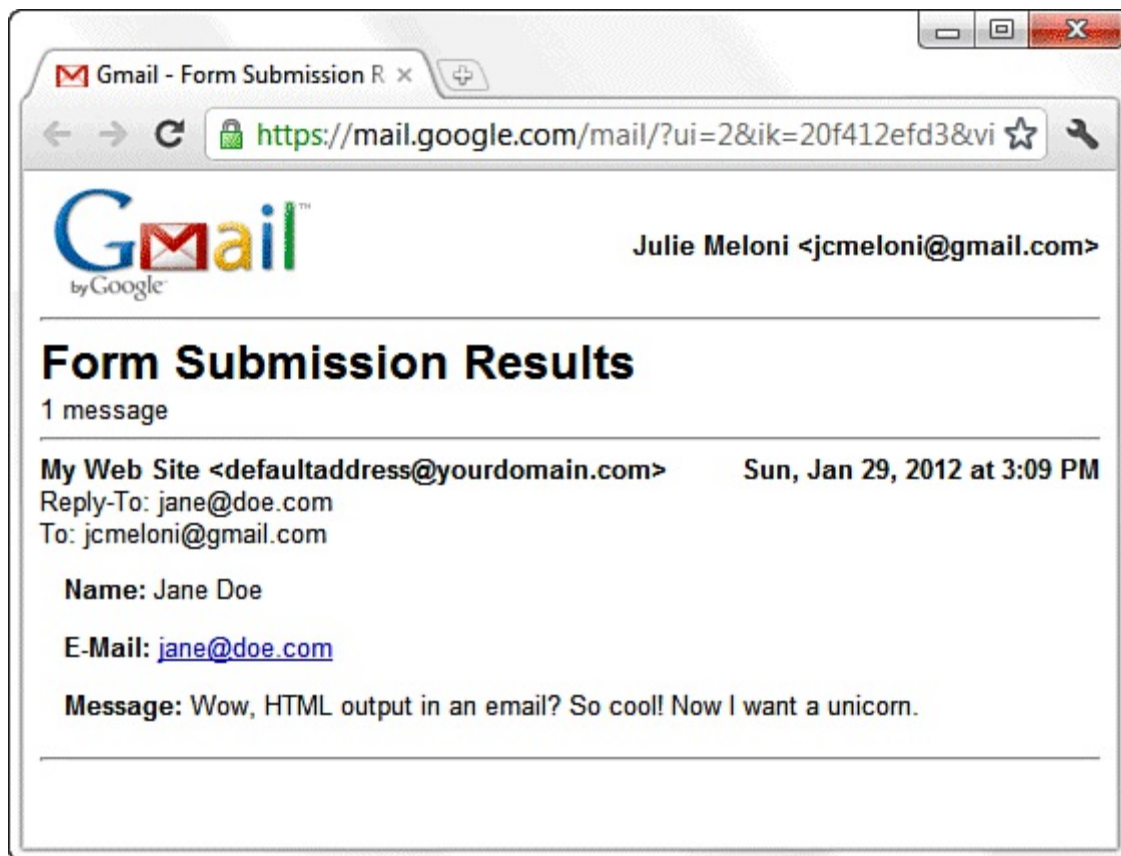


图11-8 程序清单11.13发送的E-mail

# 11.7 使用文件上传

到目前为止，我们已经看到了一些简单的表单输出。然而，Web浏览器还支持文件上传等，PHP当然也支持。在本节中，我们将学习PHP用来处理这种输入的功能。

关于上传文件的信息在\$\_FILES超全局变量中，这些信息通过表单中一个或多个上传字段的名称来索引。这些键中的每一个相应值就是一个关联数组。表11-1描述了这些字段，使用fileupload作为用于上传的表单字段的名称。

表11-1 文件上传全局变量

元 素	包 含	示 例
<code>\$_FILES["fileupload"]["name"]</code>	上传文件的最初名字	test.gif
<code>\$_FILES["fileupload"] ["tmp_name"]</code>	临时文件的路径	/tmp/phprDfZvN
<code>\$_FILES["fileupload"]["size"]</code>	上传文件的大小（字节）	6835
<code>\$_FILES["fileupload"]["type"]</code>	上传文件的MIME类型（由客户端给定）	image/gif

暂时把这些元素抛到脑后，我们将在下一节创建文件上传表单。

## 11.7.1 创建文件上传表单

首先，我们必须创建处理上传的HTML表单。包含文件上传字段的HTML表单必须包含如下一个ENCTYPE参数。

```
enctype="multipart/form-data"
```

PHP也使用一个可选的隐藏字段，这个字段可以插入到文件上传字段之前。这个字段必须叫做MAX\_FILE\_SIZE，并且应该有一个值，表示你愿意接收的文件的最大字节数。MAX\_FILE\_SIZE 字段遵循浏览器的设定，所以我们应该依赖 php.ini 设定的upload\_max\_filesize，以管控不合理的大量上传。在输入MAX\_FILE\_SIZE字段以后，我们添加了上传字段本身。这只是一个TYPE参数为“file”的INPUT元素，我们可以根据自己的意愿给它起一个名字。程序清单11.14把所有这些都写入到一个HTML上传表单中。

程序清单11.14 一个简单的文件上传表单

---

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>A simple file upload form</title>
5: </head>
6: <body>
7: <form action="do_upload.php" enctype="multipart/form-data" method="POST">
8: <input type="hidden" name="MAX_FILE_SIZE" value="1048576" />
9: <p><label for="fileupload">File to Upload:</label>
10: <input type="file" id="fileupload" name="fileupload" /></p>
11: <button type="submit" name="submit" value="send">Upload File</button>
12: </form>
13: </body>
14: </html>
```

---

正如你所看到的，在第8行，上传文件大小限制为1MB。并且上传文件字段的名称是fileupload，如第10行所示。把这个程序清单保存到



一个名为fileupload.html的文本文件中，并且将其放到Web服务器文档根目录下。使用Web浏览器访问这个表单，将看到如图11-9所示的结果。

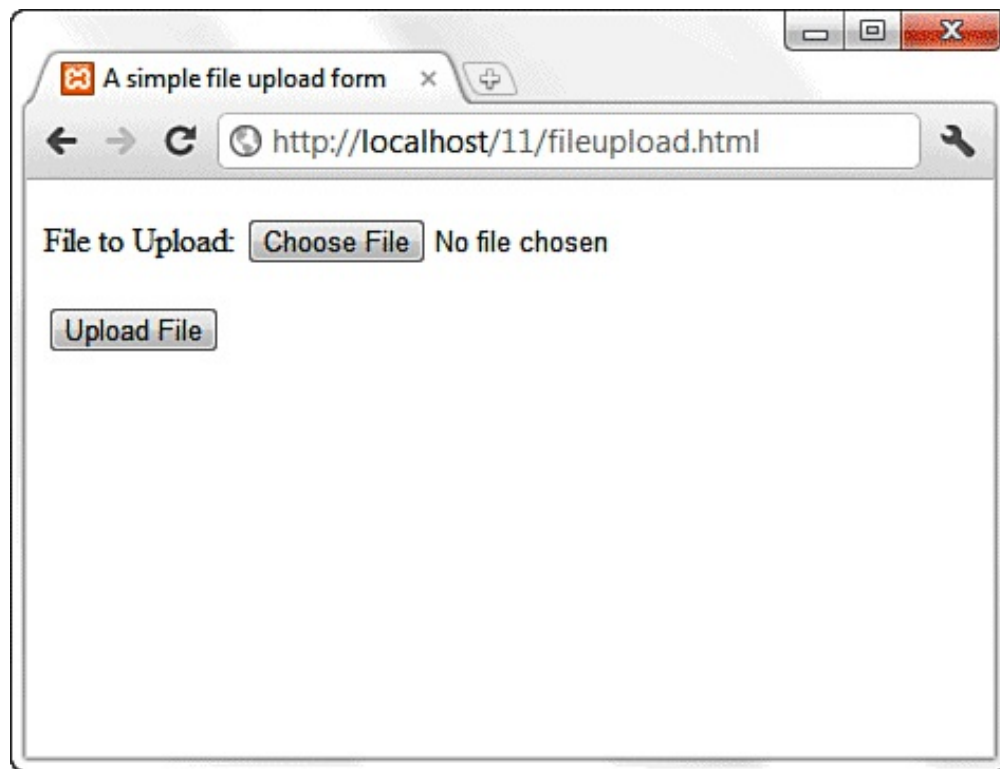


图11-9 程序清单11.14所创建的表单

这个表单调用了do\_upload.php脚本，我们随后创建这个脚本。

### 11.7.2 创建一个文件上传脚本

如果还记得有关超全局变量\$\_FILES的信息，你就具备了编写一个简单的文件上传脚本所需的所有信息。这个脚本就是程序清单11.15所创建的表单的支持代码。

程序清单11.15 一个文件上传脚本

---

```
1: <?php
2: $file_dir = "/path/to/upload/directory";
3:
4: foreach($_FILES as $file_name => $file_array) {
5:     echo "path: ".$file_array['tmp_name']."<br/>\n";
6:     echo "name: ".$file_array['name']."<br/>\n";
7:     echo "type: ".$file_array['type']."<br/>\n";
8:     echo "size: ".$file_array['size']."<br/>\n";
9:
10:    if (is_uploaded_file($file_array['tmp_name'])) {
11:        move_uploaded_file($file_array['tmp_name'],
12:            "$file_dir/".$file_array['name'])
13:        or die ("Couldn't move file");
14:        echo "File was moved!";
15:    } else {
16:        echo "No file found.";
17:    }
18: }
19: ?>
```

---

在程序清单11.15中，我们首先在第2行创建了\$`file_dir`变量来存储路径信息。这条路径应该是你的系统上存在的一条路径，并且Web服务器用户（例如，`httpd`、`www`和`nobody`）必须拥有对它的写权限。

**提示：**

第2行所使用的路径是一个Linux/UNIX路径。Windows用户应该使用转义的反斜杠，示例如下。

```
$file_dir = "C:\\Users\\You\\Desktop";
```

第4行开始一条`foreach`语句，它遍历了`$_FILES`数组中的每个元素。这里使用了一个循环而不是一条`if`语句，来使我们的脚本可以扩展到在同一个页面处理多个文件上传。第4行的`foreach`循环把上传文件名存储到`$file_name`变量中，并且把文件信息存储到`$file_array`变量中。然后，我们就可以输出所拥有的关于上传的文件信息。

在把上传文件从其临时位置移动到第2行指定的位置之前，首先检查文件是否存在。我们在第10行使用`is_uploaded_file()`函数来做这件事情。这个函数接收一条指向上传文件的路径，并且只有在所询问的文件是一个有效的上传文件的时候才返回`true`。因此，这个函数增强了脚本的安全性。

假设这些都正常，在第11行到第13行，文件将从其临时位置复制到一个新的目录，我们使用另一个函数`move_uploaded_file()`来做到这一点。这个函数把一个文件从一个位置复制到另一个位置，它首先执行和`is_uploaded_file()`函数所做的相同的安全检查。`move_uploaded_file()`函数需要一个源文件路径以及一个目标路径。如果移动成功，它返回`true`，如果文件不是有效的上传文件或者文件没有找到，它返回`false`。

注意上传文件的名称。像Mac OS和Windows这样的操作系统，它们对待文件命名问题相当宽松，因此，上传文件可能最终包括空格、引号以及各种各样其他的不可预期的字符。因此，过滤文件名是个不错的想法。

把上述代码放入到一个名为`do_upload.php`的文本文件中，并且将其放置在Web服务器文档根目录下。使用Web浏览器查看这个表单，然后尝试上传一个文件。如果成功了，你会在浏览器中看到如图11-10所示的结果。

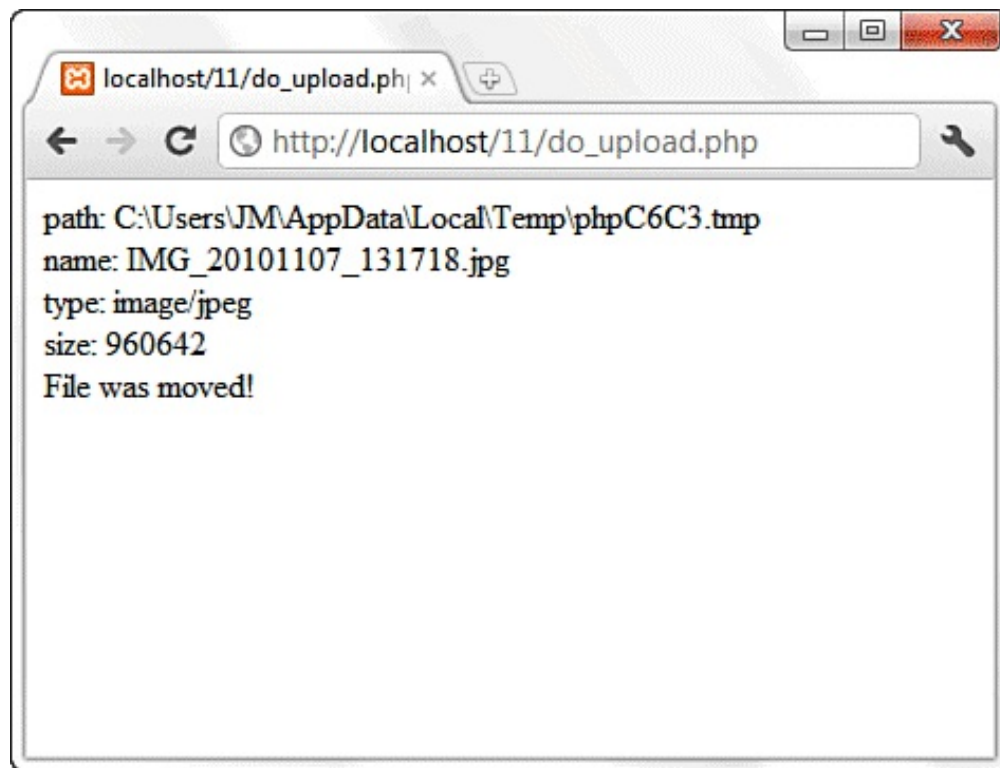


图11-10 程序清单11.14的示例结果

## 11.8 小结

事情变得令人激动起来。当然，还是漏掉了少数几项，但是本书剩下的部分将继续充实。既然已经从用户那里获取了信息，如果能够使用它们做一些事情，例如，将其写入到文件，那将会更有意思，也许是这样吧？这就是后续各章的主题。

在本章中，我们学习了如何使用各种超全局变量和表单输入。我们还学习了如何向客户端发送原始的标头以重定向用户。我们学习了如何从表单提交获取列表信息以及如何使用隐藏字段在脚本调用之间传递信息。最后，我们学习了如何在Email中发送表单结果，以及如何使用一个PHP脚本通过Web浏览器上传文件。

## 11.9 Q&A

**Q:** 当我在线提交其他表单的时候，有时候，我会看到所输入的值位于导向下一个页面的**URL**中。为什么会这样？

**A:** 如果你提交一个表单，例如一次Google搜索，并且你看到的下一个URL包含了所输入的值，例如，搜索“cheese”可能会产生如下所示的一个URL。

```
https://www.google.com/#hl=en&output=search&q=cheese
```

然后，你将会看到使用GET操作而不是POST操作的一个表单的输出。在这个例子中，至少有两个字段，一个隐藏字段，叫做“output”；另一个是可以看到的字段，叫做“q”（可能是表示query）。

“cheese”值就是你在INPUT框中输入的值。

**Q:** 为什么需要在一个表单上限制上传大小。

**A:** 如果在设计用来上传文件的表单上没有限制大小的话，你将会遇到这样的情况，就是将用户引导到一个无法完成的动作，这会导致他们的系统和你的系统被冻结。考虑下这样的情况，当你想要接受文件或数字图像上传的时候，用户创建了一个很大的图像，假设是10MB。如果你本来只是想要接受图像的缩略图，常规情况下只有350kB那么大，只要告诉用户将大小限制在这个范围就好了。只要在表单中组合MAX\_FILE\_SIZE和upload\_max\_filesize的php.ini设置，你可以确保单个

用户的操作不会阻塞网络。

## 11.10 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。



## 问答题

1. 用来找到脚本的名字的预定义变量是什么？
2. 哪个内建的关联数组包含了作为POST请求的一部分提交的所有值？
3. 哪个内建的关联数组包含了作为文件上传的一部分提交的所有值？
4. 什么函数用来把浏览器重定向到一个新的页面？
5. mail()函数所使用的4个参数是什么？
6. 在客户端，如何限制一个用户通过一个特定的上传表单所上传的文件的大小？

## 解答

1. 变量`$_SERVER['PHP_SELF']`保存了脚本的名字。
2. `$_POST`超全局变量。
3. `$_FILES`超全局变量。
4. 带有一个Location的header()函数。
5. 收件人、主题、消息字符串和附加的标头。
6. 在表单中使用一个名为`MAX_FILE_SIZE`的隐藏字段。

## 思考题

1. 创建一个计算器脚本，它允许用户提交两个数字并且选择一种对这两个数执行的运算（加减乘除）。
2. 在思考题1创建的脚本中使用隐藏字段，来存储和显示用户提交的请求的数字。

## 第12章 使用Cookie和用户会话

在本章中，你将学到：

- 如何存储和获取**cookie**信息。
- 什么是会话变量以及它们如何工作。
- 如何开始或继续一个会话。
- 如何在一个会话中存储变量。
- 如何销毁一个会话。
- 如何重新设置会话变量。

PHP 包含了很多的函数，可以用来管理和记录用户信息，包括简单的 **cookie** 和全方位的用户会话。会话使用 PHP 语言内建的技术，使得保存状态就像是引用超全局变量那样简单。

## 12.1 Cookie简介

我们可以和PHP脚本一起使用cookie来存储一些关于用户的较小的信息。cookie是由用户浏览器存储的少量数据，它和一个来自服务器或脚本的请求相一致。通过一个用户的浏览器，一个单个的主机可以请求保存20个cookie。每个cookie包含一个名字、值和过期日期，以及主机和路径信息。一个单个的cookie的大小限制是4kB。

在设置了cookie之后，只有发出请求的主机能够读取数据，这就保证了用户隐私得到尊重。另外，用户可以配置自己的浏览器通知他接受或是拒绝所有cookie的请求。因此，cookie应该适度地使用，并且在没有设计事先警告用户的一个环境中，不应该作为一个基本元素而依赖。

### 12.1.1 深入了解一个cookie

设置一个cookie的PHP脚本可能发送如下所示的标头。

```
HTTP/1.1 200 OK
Date: Wed, 18 Jan 2012 10:50:58 GMT
Server: Apache/2.2.21 (Unix) PHP/5.4.0
X-Powered-By: PHP/5.4.0
Set-Cookie: vegetable=artichoke; path=/; domain=yourdomain.com
Connection: close
Content-Type: text/html
```

正如你所看到的，Set-Cookie标头包含了一个名/值对、一个路径和一个域。如果设置了expiration字段，它会提供浏览器在哪个日期“忘记”cookie的值。如果没有设置过期日期，当用户会话过期的时候，也就是当用户关掉浏览器的时候，cookie就过期了。

path字段和domain字段协同工作，因为path是找到domain的一个目录，cookie应该送回给服务器的这个目录下面。如果路径是“/”，这是很常见的值，意味着 cookie 可以由文档根目录下的任何文件读取。如果路径是“/products/”，这个cookie只能够被Web站点的/products目录下的文件读取。

domain 字段表示允许基于 cookie 的通信所来自的 Internet 域。例如，如果我们的域是www.yourdomain.com，并且使用www.yourdomain.com作为cookie的domain值，只有在浏览www.domain.com的时候，cookie 才是有效的。如果在用户浏览的过程中，我们发送给用户诸如www2.domain.com或billing.domain.com的某个域，这可能会引发一个问题，因为最初的cookie 不再有效。因此，在 cookie 定义中的 domain 项中只是以点开头，而省略掉主机，例如.domain.com，这种方法是常见的。按照这种方法，cookie将会对该域上的所有主机都有效。域不能和cookie所发送自的域不同，否则，cookie将不能正确工作，就算是能工作，Web浏览器也会完全拒绝cookie。

### 12.1.2 访问cookies

如果Web浏览器配置为存储cookie，它将保持基于cookie的信息直到过期日期。如果用户使用浏览器浏览符合cookie的路径和域的任何页面，它将会把cookie重新发送给服务器。浏览器的标头可能会如下所示。

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like
Gecko) Chrome/16.0.912.75 Safari/535.7
```

```
Host: www.yourdomain.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en,pdf
Accept-Charset: iso-8859-1,*,utf-8
Cookie: vegetable=artichoke
```

随后，一个PHP脚本将能够访问cookie，cookie在环境变量HTTP\_COOKIE中或者作为\$\_COOKIE超全局变量的一部分，我们可以用如下3种方式来访问它们。

```
echo $_SERVER['HTTP_COOKIE']; // will print "vegetable=artichoke"
echo getenv('HTTP_COOKIE'); // will print "vegetable=artichoke"
echo $_COOKIE['vegetable']; // will print "artichoke"
```

## 12.2 使用PHP设置一个cookie

我们可以用两种方法在一个PHP脚本中设置一个cookie。首先，用header()函数来设置Set-Cookie标头。header()函数需要一个字符串，该字符串随后将包含到服务器响应的标头部分。由于标头会为你自动发送，header()必须在发送给浏览器的任何输出之前调用。

```
header("Set-Cookie: vegetable=artichoke; expires=Thu, 19-Jan-12 14:39:58 GMT; path=/; domain=yourdomain.com");
```

尽管没什么困难，这种设置cookie的方法还是需要我们编写一个函数来构建标头字符串。像这个例子那样格式化日期并对名/值对进行URL 编码并不是特别艰难的任务，但它是一项重复性的工作，因而PHP提供了一个函数，这就是setcookie()。

setcookie()函数所做的事情就像它的名字所显示的那样，它输出一个 Set-Cookie 标头。因此，它应该在任何其他内容发送给浏览器之前调用。这个函数接受cookie名字、cookie值、UNIX时间戳格式的过期日期、路径、域，以及一个整数，如果cookie仅通过一个安全的连接发送的话，这个整数的值设置为1。除了第一个参数（cookie名字）以外，这个函数的所有参数都是可选的。

程序清单12.1使用setcookie()函数设置一个cookie。

程序清单12.1 设置并显示一个cookie值



---

```
1: <?php
2: setcookie("vegetable", "artichoke", time()+3600, "/", ".yourdomain.com", 0);
3:
4: if (isset($_COOKIE['vegetable'])) {
5:     echo "<p>Hello again! You have chosen:  ".$_COOKIE['vegetable']. "</p>";
6: } else {
7:     echo "<p>Hello, you. This may be your first visit.</p>";
8: }
9: ?>
```

---

即便我们在脚本第一次运行的时候设置cookie（在第2行），`$_COOKIE [ 'vegetable' ]`变量也不会在这时候创建。由于只有当浏览器将一个cookie发送到服务器的时候，才会读取它，因此，直到用户重新访问这个域内的一个页面的时候，我们才能够读取它。

我们在第2行把cookie名字设置为“vegetable”，把cookie值设置为“artichoke”。使用`time()`函数来获取当前的时间戳，并且在其上添加3600（一小时有3600秒），这个总和表示我们的过期日期。我们定义了一个路径“/”，表示一个cookie应该为服务器环境下的所有页面发送。我们把domain参数设置为“.yourdomain.com”（你应该针对自己的域做相应的修改，或者使用localhost），这意味着cookie可以发送给该域中的任何服务器。最后，把0传递给`setcookie()`，表示cookies可以在一个非安全的环境中发送。对`setcookie()`的字符串参数发送一个空字符串或者对整数字段发送0，将会使这些参数被忽略。

给 `setcookie()` 传递一个空字符串作为字符串参数或 0 作为其整数字段参数，将会导致忽略这些参数。

#### 提示：

通过在cookie中使用一个动态创建的过期时间，如程序清单12.1所示，注意，过期时间是通过在运行Apache和PHP的机器的当前系统时间加上一定的秒数而创建的。如果这个系统时

间不是准确的，有可能它发送一个已经过去的过期时间到cookie中。

我们可以在最新的Web浏览器中查看自己的 cookie。图12-1显示了程序清单12.1中存储的cookie信息，cookie的名字、内容和过期日期都如期出现。当我们在自己的域上运行这个脚本的时候，域名可能不同。

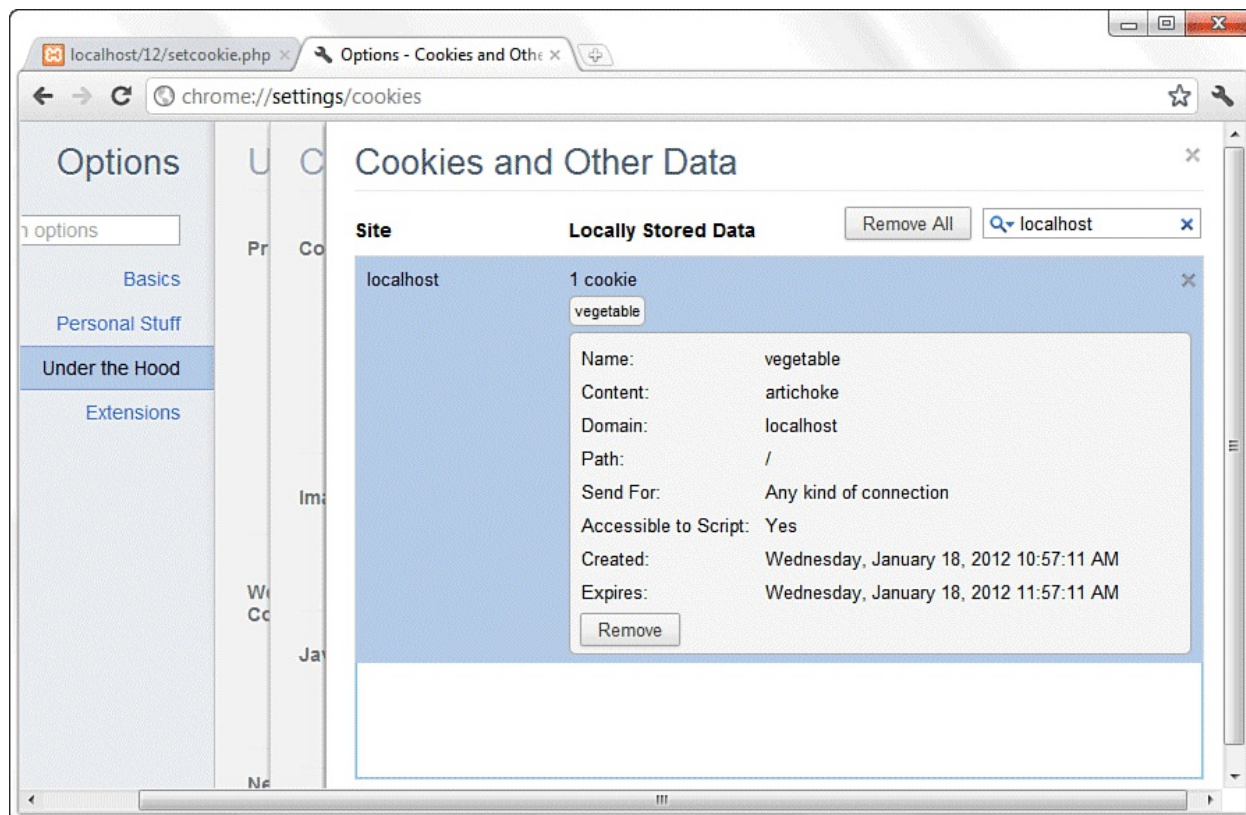


图12-1 在一个 Web 浏览器中查看存储的cookie

要了解使用cookie的更多信息，尤其是setcookie()函数的信息，请参考位于<http://www.php.net/setcookie>的PHP手册。

## 删除一个cookie

正式地讲，要删除一个 cookie，只需要调用带有 cookie 名字参数的 setcookie()，示例如下。

```
setcookie("vegetable");
```

然而，这种方法并不总是奏效，并且不能依赖这种方法。相反，要删除一个cookie，使用一个确定已经过期的时间来设置cookie，这种方法是最安全的。

```
setcookie("vegetable", "", time()-60, "/", ".yourdomain.com", 0);
```

还要确保传递给setcookie()与最初设置cookie时候所使用的是相同的路径、域和安全参数。

## 12.3 会话函数概览

会话函数为用户提供了一个唯一的标识符，随后可以用来存储和获取连接到该标识符的信息。当一个访客访问一个支持会话的页面，要么分配一个新的标识符，要么这个用户和之前的访问已经建立的一个标识符重新关联。任何已经和会话相关联的变量，都通过`$_SESSION` 超全局变量变得可供你的代码使用。

会话状态通常存储在一个临时文件中，尽管你可以使用一个名为`session_set_save_handler()`的函数实现数据库存储。`session_set_save_handler()`函数的使用以及有关其他高级会话功能的讨论已经超出了本书的范围，但是，你可以在PHP手册关于会话的部分找到这里所没有讨论的项目的所有信息。

## 12.4 开始一个会话

要使用一个会话，我们需要显式地开始或继续会话，除非我们已经改变了php.ini配置文件。默认情况下，会话不会自动启动。如果想要按这种方法启动一个会话，我们必须在php.ini文件中找到如下的一行，将其值从0改为1，并且重新启动Web服务器。

```
session.auto_start = 0
```

通过把session.auto\_start的值改为1，我们可以确保一个会话针对每个PHP文档而启动。如果你不想改变这一设置，需要在每个脚本中调用session\_start()函数。

会话启动之后，我们立即可以通过 session\_id()函数访问用户的会话ID，session\_id ()函数允许你设置或访问一个会话ID。程序清单 12.2 开始一个会话并且将会话ID输出到浏览器。

程序清单12.2 开始或继续一个会话

---

```
1: <?php
2: session_start();
3: echo "<p>Your session ID is ".session_id()."</p>";
4: ?>
```

---

当这个脚本第一次从一个浏览器运行的时候，在第2行的session\_start()函数调用产生一个会话ID。如果这个脚本稍后重新载入或者重新访问，同一个会话ID也分配给该用户。这个操作假设该用户已经可以使用cookie了。例如，当我们第一次运行这个脚本的时候，输出如下。

Your session ID is 8jou17in51d08e5onsjkbles16.

当我们重新载入这个页面的时候，输出仍然如下。

Your session ID is 8jou17in51d08e5onsjkbles16.

因为我们已经可以使用cookie了并且会话ID仍然存在。

当第一次初始化一个会话的时候，由于start\_session()尝试设置一个cookie，因此必须在向浏览器输出任何内容之前调用这个函数。如果没有遵守这一规则，会话就不会被设置，并且你将可能在页面上看到警告。

只要 Web 浏览器是激活的，就将一直保持当前会话。如果用户重新打开浏览器，cookie就不再存储。我们可以在php.ini文件中修改session.cookie\_lifetime设置，从而改变这一行为。默认值是0，但是我们可以设置一个以秒为单位的过期周期。

## 12.5 使用会话变量

在每一个PHP文档中访问一个唯一的会话标识符只是会话功能的开始。当一个会话启动后，我们可以在超全局变量\$\_SESSION中存储任意多个变量，然后在任何支持会话的页面上访问它们。

程序清单12.3向超全局变量\$\_SESSION添加了两个变量：product1和product2（第3行和第4行）。

程序清单12.3 在一个会话中存储变量

---

```
1: <?php
2: session_start();
3: $_SESSION['product1'] = "Sonic Screwdriver";
4: $_SESSION['product2'] = "HAL 2000";
5: echo "The products have been registered.";
6: ?>
```

---

在用户移动到另一个新的页面之前，程序清单12.3中的神奇之处不会体现出来。程序清单12.4创建了一个单独的PHP脚本，这个脚本访问存储在超全局变量\$\_SESSION中的变量。

程序清单12.4 访问存储的会话变量

---

```
1: <?php
2: session_start();
3: ?>
4: <p>Your chosen products are:</p>
5: <ul>
6: <li><?php echo $_SESSION['product1']; ?></li>
7: <li><?php echo $_SESSION['product2']; ?></li>
8: </ul>
```

---

图12-2 显示了来自程序清单 12.4 的输出。正如你所见到的，我们

已经在一个全新的页面中访问了\$\_SESSION [ 'product1' ]和\$\_SESSION [ 'product2' ]变量。

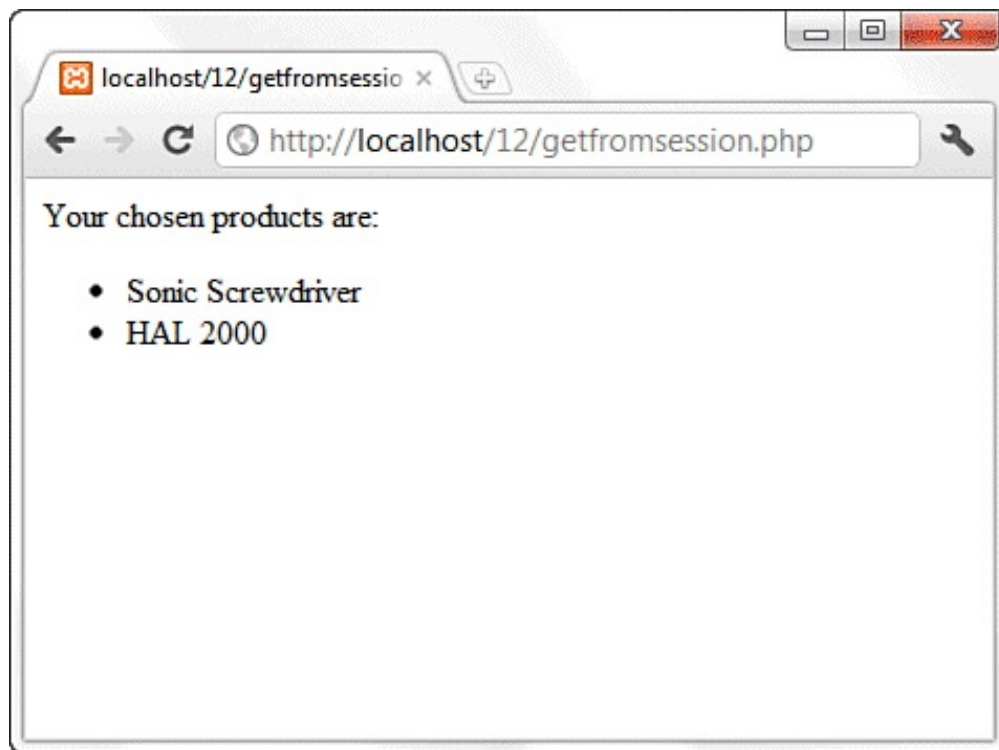


图12-2 访问存储的会话变量

尽管不是一个非常有趣或有用的例子，但这个脚本确实展示了如何访问存储的会话变量。在幕后，PHP把信息写入到一个临时文件，可以使用`session_save_path()`函数查看这个文件被写到了系统上的什么地方。这个函数可选地接受到一个目录的路径，并且把所有的会话文件都写入其中。如果我们不传递给它参数，它返回一个字符串，表示会话文件保存的当前目录。在我的系统上执行如下语句。

```
echo session_save_path();
```

会显示出一个/tmp。看一下我的/tmp目录会显示出很多文件，其名字如下所示。



```
sess_fa963e3e49186764b0218e82d050de7b  
sess_76cae8ac1231b11afa2c69935c11dd95  
sess_bb50771a769c605ab77424d59c784ea0
```

当我第一次运行程序清单 12.2 的时候，打开和所分配的会话 ID 相匹配的文件，可以看到已注册的变量是如何存储的。

```
product1|s:17:"Sonic Screwdriver";product2|s:8:"HAL 2000";
```

当一个值放置在\$\_SESSION 超全局变量中，PHP把变量名和值写入到一个文件中。正如我们所看到的，这个信息可以读取并且变量可以稍后恢复。当我们把这个变量添加到超全局变量\$\_SESSION后，你仍然可以在脚本执行过程中的任何时刻修改其值，但是，这个修改后的值不会反映到全局设置中，直到把这个变量重新赋值给超全局变量\$\_SESSION。

程序清单12.3中的例子展示了把变量添加到超全局变量\$\_SESSION的过程。然而，这个例子并不是非常灵活。理想情况下，你所能注册的值的数目应该是可变的。例如，可能希望用户从一个列表中选择产品。在这种情况下，可以使用serialize()函数来把数组存储到会话中。

程序清单12.5创建了一个表单，它允许一个用户来选择多个产品。我们可以使用会话变量来创建一个基本的购物车。

程序清单12.5 把一个数组变量添加到一个会话变量中

---

```
1: <?php
2: session_start();
3: ?>
4: <!DOCTYPE html>
5: <html>
6: <head>
7: <title>Storing an array with a session</title>
8: </head>
9: <body>
10: <h1>Product Choice Page</h1>
11: <?php
12: if (isset($_POST['form_products'])) {
13:     if (!empty($_SESSION['products'])) {
14:         $products = array_unique(
15:             array_merge(unserialize($_SESSION['products']),
16:                 $_POST['form_products']));
17:         $_SESSION['products'] = serialize($products);
18:     } else {
19:         $_SESSION['products'] = serialize($_POST['form_products']);
20:     }
21:     echo "<p>Your products have been registered!</p>";
22: }
23: ?>
24: <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
25: <p><label for="form_products">Select some products:</label><br />
26: <select id="form_products" name="form_products[]" multiple="multiple"
    size="3">
27: <option value="Sonic Screwdriver">Sonic Screwdriver</option>
28: <option value="Hal 2000">Hal 2000</option>
29: <option value="Tardis">Tardis</option>
30: <option value="ORAC">ORAC</option>
31: <option value="Transporter bracelet">Transporter bracelet</option>
32: </select></p>
33: <button type="submit" name="submit" value="choose">Submit Form</button>
34: </form>
35: <p><a href="session1.php">go to content page</a></p>
36: </body>
37: </html>
```

---

我们通过在第2行调用`session_start()`来启动或继续一个会话。这个调用使我们能够访问之前设置的任何会话变量，第24行开始一个HTML表单，在第26行创建了一个名为`form_products []`的SELECT元素，其中包含了多个产品的OPTION元素。

提示：

别忘了，HTML的表单元素允许拥有方括号的多个选项附加到它们的NAME参数的值之后。这就使得用户的选择可以在一个数组中使用。

在第11行开始的PHP代码块中，我们测试了`$_POST ['form_products']`数组的存在（第 12 行）。如果这个变量存在，我们可以假设表单已经提交并且信息已经存储到超全局变量`$_SESSION`中。

随后，在第12行测试一个叫做`$_SESSION ['products']`的数组。如果这个数组存在，之前访问这个脚本的时候填充过该数组，这样我们就可以将其和`$_POST ['form_products']`数组合并，提取出唯一的元素，并且把结果赋值回`$products`数组（第14行到第16行）。随后，在第17行把`$products`数组添加到`$_SESSION`超全局变量。

第 35 行包含了到其他脚本的一个链接，我们将用这个脚本展示对用户选取的产品的访问。我们在程序清单 12.6 中创建了这个脚本，同时把程序清单 12.5 中的代码保存为`arraysession.php`。

来看看程序清单12.6是如何访问存储到会话中的那些项的，注意会话是在`arraysession.php`中创建的。

程序清单12.6 访问会话变量

---

```
1: <?php
2: session_start();
3: ?>
4: <!DOCTYPE html>
5: <html>
6: <head>
7: <title>Accessing session variables</title>
8: </head>
9: <body>
10: <h1>Content Page</h1>
11: <?php
12: if (isset($_SESSION['products'])) {
13:     echo "<strong>Your cart:</strong><ol>";
14:     foreach (unserialize($_SESSION['products']) as $p) {
15:         echo "<li>".$p."</li>";
16:     }
17:     echo "</ol>";
18: }
19: ?>
20: <p><a href="arraysession.php">return to product choice page</a></p>
21: </body>
22: </html>
```

---

再一次，我们在第 2 行使用`session_start()`来继续会话，在第 12 行测试了`$_SESSION['products']`变量的存在性。如果存在，就会将它反序列化，并且在第 14 行到第 16 行遍历它，在浏览器显示出每个用户的选择项。图 12-3 给出了这个例子的运行效果。

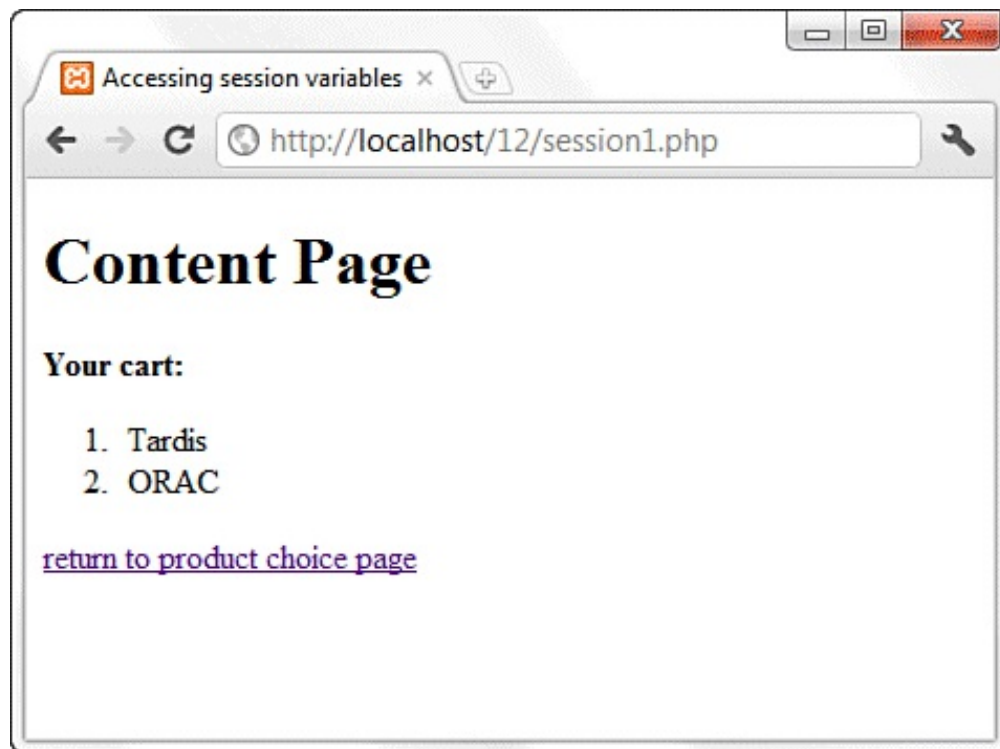


图12-3 访问会话变量的一个数组

当然，对于一个真正的购物车程序，我们应该把产品细节保存在一个数据库中并且测试用户输入，而不是盲目地存储和显示它。但是，程序清单12.5和程序清单12.6展示了我们可以很容易地使用会话函数来访问其他页面中设置的数组变量。

## 12.6 销毁会话和重置变量

我们可以使用`session_destroy()`来结束一个会话，消除所有的会话变量。`session_destroy()`函数不需要参数，应该有一个已经建立的会话供这个函数操作。如下的代码段首先继续一个会话然后销毁它。

```
session_start();  
session_destroy();
```

当我们移动到使用一个会话的其他页面的时候，已经销毁的会话将不能再使用，这迫使它们启动自己的新的会话。任何已注册的变量将会丢失。

然而，`session_destroy()`函数不会立刻销毁已注册的变量。对于那些在其中调用了`session_destroy()`的脚本，它们仍然可以访问这些变量直到变量被重新载入。下面的代码段首先继续或者启动一个会话，并且注册一个名为`test`的变量，这个变量的值我们设置为5。销毁这个会话并不会销毁这个已注册的变量。

```
session_start();  
$_SESSION['test'] = 5;  
session_destroy();  
echo $_SESSION['test']; // prints 5
```

要从一个会话中删除所有已注册变量，只需要简单地重置变量，如下所示。

```
session_start();  
$_SESSION['test'] = 5;  
session_destroy();  
unset($_SESSION['test']);  
echo $_SESSION['test']; // prints nothing (or a notice about an undefined index)
```

## 12.7 在一个带有注册用户的环境中会话

到现在所见到的例子对于会话还都只是浅尝辄止，但是，可以这么说，可能需要一些附加说明来防止会话的“滥用”。下面的两个小节给出了常见会话用法的一些例子。在本书稍后的各章中，会话将用于我们将要构建的一个示例应用程序。

### 12.7.1 使用注册的用户

假设你已经创建了一个在线社区或门户网站，或者是用户可以加入的某种其他类型的应用，这个过程通常包含一个注册表单，用户在其中创建一个用户名和密码并且填写一个个人身份表格。从这个页面发送的那一刻开始，每次一个注册的用户登录到系统的时候，我们都可以抓取用户的身份信息并且将其存储到一个用户会话中。

你决定要存储到一个用户会话中的项目应该是这样的项，我们可以想象得到，这些项会用得很少，并且不断地从数据库中提取它们的效率很差。例如，假设我们创建了一个门户网站，其中的用户会分配一个级别，例如管理员、注册用户、匿名访客等等。在我们的显示模式中，应该总是要检查验证用户所访问的模块是否对他有相应的许可。因此，“用户级别”应该是存储在用户会话中的一个值，所以，在显示请求的模块中所用到的身份验证脚本只需要去检查一个会话变量，就没必要连接到数据库并查询数据了。

### 12.7.2 使用用户偏好

如果你想在一個基於用戶的應用程序的设计阶段感受新潮，可以构建这样一个系统，让注册用户可以设置具体的偏好来影响他浏览你的站点的方式。例如，可以允许用户从一个预先确定的颜色方案、字体字号等等中做出选择。或者，我们可能会允许用户打开或关闭对于某一组内容的浏览。

这些功能元素中的每一个都应该存储到一个会话中。当用户登录，这个应用程序应该把所有相关的值都载入到该用户的会话中，并且据此来对后续请求的每一个页面做出反应。这个用户可以修改其偏好，他可以在登录的时候做到这一点，我们甚至应该根据存储在会话中的项目来预先装载一个“偏好”表单，而不是回到数据库来获取这些项目。如果该用户在登录的时候修改了任何偏好，只要使用新的选项来替换 `$_SESSION` 超全局变量中的值就可以了，不需要迫使用户退出然后再次重新登录。



## 12.8 小结

在本章中，我们学习了在一个无状态的协议中保存状态的不同方法，包括设置一个cookie和启动一个会话。所有这些保存状态的方法都使用了某种方式的cookie或者查询字符串，有时候会结合使用到文件或数据库。这些方法都各有优缺点。

我们了解了一个cookie本质上不是可靠的，并且不能存储太多的信息。另一方面，它可以维持一个较长的时间。把信息写入文件或数据库将导致速度的降低，并且在一个公共的站点上可能会有问题，这是和系统管理相关的一个问题。

关于会话自身，我们学习了如何使用`session_start()`启动或继续一个会话。当处于一个会话中时，我们学习了如何向`$_SESSION`超全局变量添加变量，检查其存在性，如果需要的话重置它们，以及销毁整个会话。

## 12.9 Q&A

**Q:** 如果用户不能够使用**cookie**的话，应用程序将会发生什么情况？

**A:** 简而言之，如果应用程序较强地依赖于 **cookie**并且用户禁用 **cookie**，应用程序将无法工作。然而，可以通过声明你要使用 **cookie**来告诉用户开启**cookie**；并且，在对应用程序做任何“重要的”事情之前，也要检查 **cookie**是否可以使用。当然，这么做的意图是，即便用户忽略了你为了使用应用程序必须打开**cookie**的提示，如果**cookie**测试失效了而导致不允许用户执行一个操作，这也会唤起用户的注意。

**Q:** 我应该知道到会话函数的所有缺点吗？

**A:** 会话函数通常是可靠的。然而，别忘了，**cookie** 不能跨越多个域读取，因此，如果项目在同一个服务器上使用多个域名（可能作为一个电子商务环境的一部分），你可能需要考虑通过在**php.ini**文件中把 **session.use\_cookies**指令设置为0来关闭会话的**cookie**。

## 12.10 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 哪个函数用来以一个PHP脚本启动或继续一个会话？
2. 哪个函数可以返回当前会话的ID？
3. 如何终结一个会话并且删除对其访问的所有记录？

## 解答

1. 我们可以在脚本中使用`session_start()`函数来启动一个会话。
2. 我们可以使用`session_id()`函数来访问会话的ID。
3. `session_destroy()`函数删除一个会话及请求的所有记录。

## 思考题

1. 创建一个脚本，它使用会话函数来记录在你的环境中用户访问了哪些页面。
2. 创建一段新的脚本，它整页整页地列出在你的环境中访问的用户，以及他们访问的时间。

## 第13章 使用文件和目录

在本章中，你将学到：

- 如何把其他文件包含到你的文档中。
- 如何测试文件和目录的存在性。
- 如何在使用文件前打开它。
- 如何从文件读取数据。
- 如何写入一个文件或向其添加内容。
- 如何锁定一个文件。
- 如何使用目录。
- 如何把数据从外部应用程序导入和导出。
- 如何发送**shell**命令并且在浏览器显示结果。

测试文件的存在性以及读写文件，对于富编程语言来说是很重要的功能。PHP 也不例外，因此，它提供了很多函数使得这一过程相当简单。另外，由于PHP、Apache和MySQL不是机器上仅有的应用程序，我们也可能需要在PHP代码中访问其他的应用程序。本章中的元素向你展示如何从PHP代码访问其他的应用程序。

## 13.1 使用include语句包含文件

include语句使得我们能够把其他文件，通常是其他PHP脚本包含到PHP文档中。这些被包含的文件中的PHP代码就好像是主文档的一部分一样来执行，这对于在多个页面中包含代码库是很有用的。

即便我们创建了一个真正有用的函数，如果没有include语句，我们到现在还只能选择将函数粘贴到需要使用它的每个文档中。如果发现函数有一个漏洞（bug）或者想要添加一项功能，我们必须找到将其贴入的每个页面并且修改它，一次又一次地重复。Include语句使我们不必这么繁琐。我们可以把新创建的函数添加到一个单独的文档中，例如myinclude.php，然后，在运行的时候，将其读入到需要它的任何页面。

Include语句需要一个参数：指向要包含的文件的相对路径。程序清单 13.1 创建了一个简单的PHP脚本，这个脚本使用include语句来包含一个文件，并且输出该文件的内容。

程序清单13.1 使用include()

---

```
1: <?php
2: include 'myinclude.php';
3: ?>
```

---

程序清单13.1中的include语句包含了文档myinclude.php，我们可以在程序清单 13.2 中看到被包含文档的内容。

程序清单13.2 被包含到程序清单13.1中的文档

---

```
1: I have been included!!
```

---



把程序清单 13.1的内容放入到一个名为test\_include.php的文件，并且把程序清单 13.2的内容放入到一个名为myinclude.php的文件。把这两个文件都放到Web服务器文档根目录下，当我们使用Web浏览器访问test\_include.php的时候，将会输出如下信息。

```
I have been included!!
```

我们在一个PHP代码块中包含了纯文本，这个结果对你来说看上去有些奇怪。实际上，一个被包含文件的内容默认显示为文本。如果想要在一个被包含文件中执行PHP代码，必须使用PHP开始和结束标记将其包围起来。在程序清单13.3中，我们修改了myinclude.php的内容，以便代码在一个被包含文件中执行。

程序清单13.3 包含PHP代码的一个包含文件

---

```
1: <?php
2: echo "I have been included!!<br/>";
3: echo "But now I can add up... 4 + 4 = " . (4 + 4);
4: ?>
```

---

把程序清单 13.3 的内容放入到一个名为 myinclude2.php 的文件中，并且修改包含在test\_include.php中的被包含文件的名，让它指向这个新的文件。把这些文件都放到Web服务器文档根目录下。现在，当我们使用Web浏览器来访问test\_include.php的时候，输出如下所示。

```
I have been included!!
But now I can add up... 4 + 4 = 8
```

我们能看到数字8的唯一途径是，把4和4相加的代码作为PHP执行，并且事实正是如此。

### 13.1.1 从一个被包含文档返回一个值

PHP中的被包含文件可以像函数一样返回一个值。就像在函数中一样，在被包含文件的末尾使用return语句来结束代码的执行。此外，不包含额外的HTML。在程序清单 13.4 和程序清单 13.5中，我们包含了一个文件并且将其返回值赋给一个变量。

程序清单13.4 使用include来执行PHP并且赋返回值

---

```
1: <?php
2: $addResult = include 'returnvalue.php';
3: echo "The include file returned ".$addResult;
4: ?>
```

---

程序清单13.5 一个返回一个值的包含文件

---

```
1: <?php
2: $retval = ( 4 + 4 );
3: return $retval;
4: ?>
5: This HTML will never be displayed because it comes after a return statement!
```

---

把程序清单13.4的内容放入到一个名为test\_returnvalue.php的文件中，把程序清单 13.5 的内容放入到一个名为returnvalue.php的文件中，把这些文件都放入到Web服务器文档根目录中。当我们通过Web浏览器访问test\_returnvalue.php的时候，输出如下所示。

The include file returned 8.

正如程序清单13.5第5行的字符串所示，如果在被包含文件中有一个PHP语句块的话，PHP语句块以外的任何内容都不会显示。

### 13.1.2 在控制结构中使用include语句

如果我们想要在一个条件语句中使用include语句，它将当作任何其他代码一样处理，只有条件满足的时候，才会包含文件。例如，如下代

码段中的include语句就不会调用。

```
$test = "1";  
if ($test == "2") {  
    include 'file.txt'; // won't be included  
}
```

如果我们在一个循环中使用一条include语句，那么每次循环迭代的时候都将包含文件一次。程序清单13.6通过在一个for循环中使用一条include语句说明了这一概念。对于每次迭代，include语句都引用一个不同的文件。

程序清单13.6 在一个循环中使用include

---

```
1: <?php  
2: for ($x = 1; $x<=3; $x++) {  
3:     $incfile = "incfile".$x.".txt";  
4:     echo "Attempting to include ".$incfile."<br/>";  
5:     include $incfile;  
6:     echo "<hr/>";  
7: }  
8: ?>
```

---

把程序清单13.6的内容保存到一个名为loopy\_include.php的文件中，并且将其放置到Web服务器的文档根目录下，同时一起放置的还有其他3个不同的文件incfile1.txt、incfile2.txt和incfile3.txt。假设这些文件中的每一个只是简单地包含自己的文件名的一个确认，那么输出将会如图13-1所示。



图13-1 loopy\_include.php的输出

尽管这段代码工作得很好，但一些情况下，按照这种方式使用include语句并不是一个好的想法，我们将在下一节中看到。

### 13.1.3 使用include\_once语句

在代码中使用几个不同的代码库所引起的问题之一，就是存在同一个文件被两次调用include的风险。这有时候会发生在较大的项目上，即当不同的库文件在一个共同的文件上调用include的时候。两次包含同一个文件，常常会导致重复声明函数和类，因而引发PHP引擎的异常。

通过使用include\_once语句来代替include语句可以补救这种情况。include\_once语句需要一个包含文件的路径，此外，第一次调用它的情

况和include语句是完全相同的。然而，如果在脚本执行中对同一个文件再次调用include\_once语句，这个文件不会被再次包含。这使得include\_once语句成为创建可重用的代码库的一个优秀工具。

### 13.1.4 include\_path命令

使用include()和include\_once()来访问代码库可以增加项目的灵活性和可重用性。然而，还是有一些头痛的问题需要克服。例如可移植性，如果你直接编码包含的文件的路径，将会为此痛苦不堪。假设创建了一个lib目录并且通过自己的项目以如下方式来引用它。

```
include_once '/home/user/bob/htdocs/project4/lib/mylib.inc.php';
```

当把自己的项目移动到一个新的服务器时，如果包含路径在一百个甚至更多的文件中都是直接编码的话，你可能会发现必须去修改一百个甚至更多的包含路径。可以通过在php.ini文件中设置include\_path命令来避免这一情况。

```
include_path './home/user/bob/htdocs/project4/lib/
```

include\_path的值可以包含任意多的目录，中间用冒号隔开（在Windows中是分号）。include\_path命令中项目的顺序决定了查找指定的文件目录的顺序。第一个冒号前的第一个点(.)表示“当前目录”，默认应该给出。现在，在include或include\_once中使用的任何路径都是相对于include\_path的值，示例如下。

```
include_once 'mylib.inc.php';
```

当转到自己的项目时，你只需要修改include\_path命令。

提示：

PHP 也有一个require语句，它执行和include语句类似的功能，另外还有一个require\_once语句。不管脚本的流程如何，require语句所带入到代码中的一切内容都会执行，因此，不应该将其用作条件或循环结构的一部分。另外要注意，作为一条require语句的结果所包含的文件不能返回一个值。

## 13.2 验证文件

在代码中使用一个文件或目录之前，对它有个详细了解是一个不错的想法，并且知道它是否真的存在是一个不错的开始。PHP提供了很多的函数来帮助发现关于你的系统上的文件的信息。本节简短地介绍一些最有用的函数。

### 13.2.1 使用file\_exists()检查文件的存在性

我们可以使用file\_exists()函数测试一个文件的存在性。这个函数需要表示一个文件的绝对路径或者相对路径的一个字符串，该文件可能存在也可能不存在。如果找到该文件，file\_exists()函数返回true；否则它返回false。

```
if (file_exists('test.txt')) {  
    echo "The file exists!";  
}
```

这些都没错，但是，如果你不能确定这是一个文件或者一个目录，并且真的需要知道，那该怎么办呢？

### 13.2.2 文件还是目录

我们可以使用is\_file()函数来确定所要测试的实体是一个文件，或者是一个目录。is\_file()函数需要文件的路径并且会返回一个布尔值。

```
if (is_file('test.txt')) {  
    echo "test.txt is a file!";  
}
```

相反，我们可能想要检查所要测试的是否是一个目录，可以使用

`is_dir()`函数来做到这一点。`is_dir()`需要目录的路径作为参数并且返回一个布尔值。

```
if (is_dir('/tmp')) {  
    echo "/tmp is a directory";  
}
```

当我们知道一个文件或目录存在后，可能需要测试其是否允许访问。我们将在下一节学习这一点。

### 13.2.3 检查一个文件的状态

当知道一个特定的实体存在，并且它是我们所期望的一个目录或者一个文件，我们将需要知道可以对它做什么。通常，你可能希望读取、写入或执行一个文件。**PHP**可以帮助你确定是否可以执行这些操作。

`is_readable()`函数告诉你是否可以读取一个文件。在UNIX系统上，我们或许能够看到一个文件，但是仍然禁止读取其内容，因为需要用户许可。`is_readable()`函数接受一个字符串作为文件路径，并且返回一个布尔值。

```
if (is_readable('test.txt')) {  
    echo "test.txt is readable";  
}
```

`is_writable()`函数告诉你是否拥有写入一个文件的许可。和`is_readable()`一样，`is_writable()`函数需要一个文件路径并且返回一个布尔值。

```
if (is_writable('test.txt')) {  
    echo "test.txt is writable";  
}
```



`is_executable()`函数告诉我們是否可以根据给定文件的许可或者其扩展名，在你的特定平台上执行它。这个函数接受文件的路径并且返回一个布尔值。

```
if (is_executable('test.txt')) {  
    echo "test.txt is executable";  
}
```

和许可相关的信息并不是我们需要知道的关于一个文件的全部信息。下一节将展示如何确定文件的大小。

### 13.2.4 使用**filesize()**确定文件的大小

给定了文件的路径，`filesize()`函数尝试确定并返回文件的大小（以字节为单位）。如果遇到问题，它返回`false`，示例如下。

```
echo "The size of test.txt is ".filesize('test.txt');
```

当我们想要把一个文件作为一封Email的附件或者想要把文件传送给用户的时候，知道具体文件的大小是很重要的，知道文件的大小，在Email的情况下以便正确地创建标头或者在流传送的情况下知道何时停止向用户发送字节。对更通用的情况而言，我们可能想获取文件的大小，以便可以在用户试图从你的站点下载超大容量的应用程序或者清晰度的照片之前，将文件大小显示给他们。

### 13.2.5 获取有关一个文件的日期信息

有时候，我们需要知道一个文件最后一次写入或访问的时间。PHP提供的几个函数可以提供这些信息。

我们可以使用`fileatime()`函数来获取一个文件的最后访问时间。这

个函数需要文件的路径并且返回最后访问文件的日期。要访问一个文件，意味着读取它或者写入它。从所有日期信息函数返回的日期都是时间戳的形式，也就是说，从1970年1月1日起的秒数。在我们的例子中，使用date()函数来把这个值转换为人们可以识别的形式，示例如下。

```
$atime = fileatime("test.txt");  
echo "test.txt was last accessed on ".date("D d M Y g:i A", $atime);  
// Sample output: test.txt was last changed on Sat 26 Apr 2008 12:52 PM
```

我们可以使用filemtime()函数来获得一个文件的修改日期。这个函数需要一个文件路径，并且以UNIX时间戳格式返回日期。要修改一个文件，意味着以某种方式修改其内容，示例如下。

```
$mtime = filemtime("test.txt");  
echo "test.txt was last modified on ".date("D d M Y g:i A", $mtime);  
// Sample output: test.txt was last modified on Wed 18 Jan 2012 7:11 PM
```

PHP 也允许我们使用 filectime()函数测试一个文件的修改时间。在UNIX 系统上，当一个文件的内容被修改或者当其许可权限或所有者发生变化的时候，就设置修改日期。在其他的平台上，filectime()函数返回文件的创建日期，示例如下。

```
mtime = filemtime("test.txt");  
echo "test.txt was last modified on ".date("D d M Y g:i A", $mtime);  
// Sample output: test.txt was last modified on Wed 18 Jan 2012 7:11 PM
```

### 13.2.6 编写一个执行多文件测试的函数

程序清单13.7创建了一个函数，它把我们刚刚介绍的一些和文件相关的函数引入到一个脚本中。

程序清单13.7 输出多个文件测试的结果的一个函数

---

```
1: <?php
2: function outputFileTestInfo($f) {
3:     if (!file_exists($f)) {
4:         echo "<p>$f does not exist</p>";
5:         return;
6:     }
7:     echo "<p>$f is ".(is_file($f) ? "" : "not ")."a file</p>";
8:     echo "<p>$f is ".(is_dir($f) ? "" : "not ")."a directory</p>";
9:     echo "<p>$f is ".(is_readable($f) ? "" : "not ")."readable</p>";
10:    echo "<p>$f is ".(is_writable($f) ? "" : "not ")."writable</p>";
11:    echo "<p>$f is ".(is_executable($f) ? "" : "not ")."executable</p>";
12:    echo "<p>$f is ".(filesize($f))." bytes</p>";
13:    echo "<p>$f was accessed on ".date( "D d M Y g:i A",filemtime($f))."</p>";
14:    echo "<p>$f was modified on ".date( "D d M Y g:i A",filemtime($f))."</p>";
15:    echo "<p>$f was changed on ".date( "D d M Y g:i A",filemtime($f) )."</p>";
16: }
17: $file = "test.txt";
18: outputFileTestInfo($file);
19: ?>
```

---

如果这段代码作为 filetests.php 保存到 Web 服务器的文档根目录下，并且通过 Web 浏览器来运行，输出将会如图13-2所示。

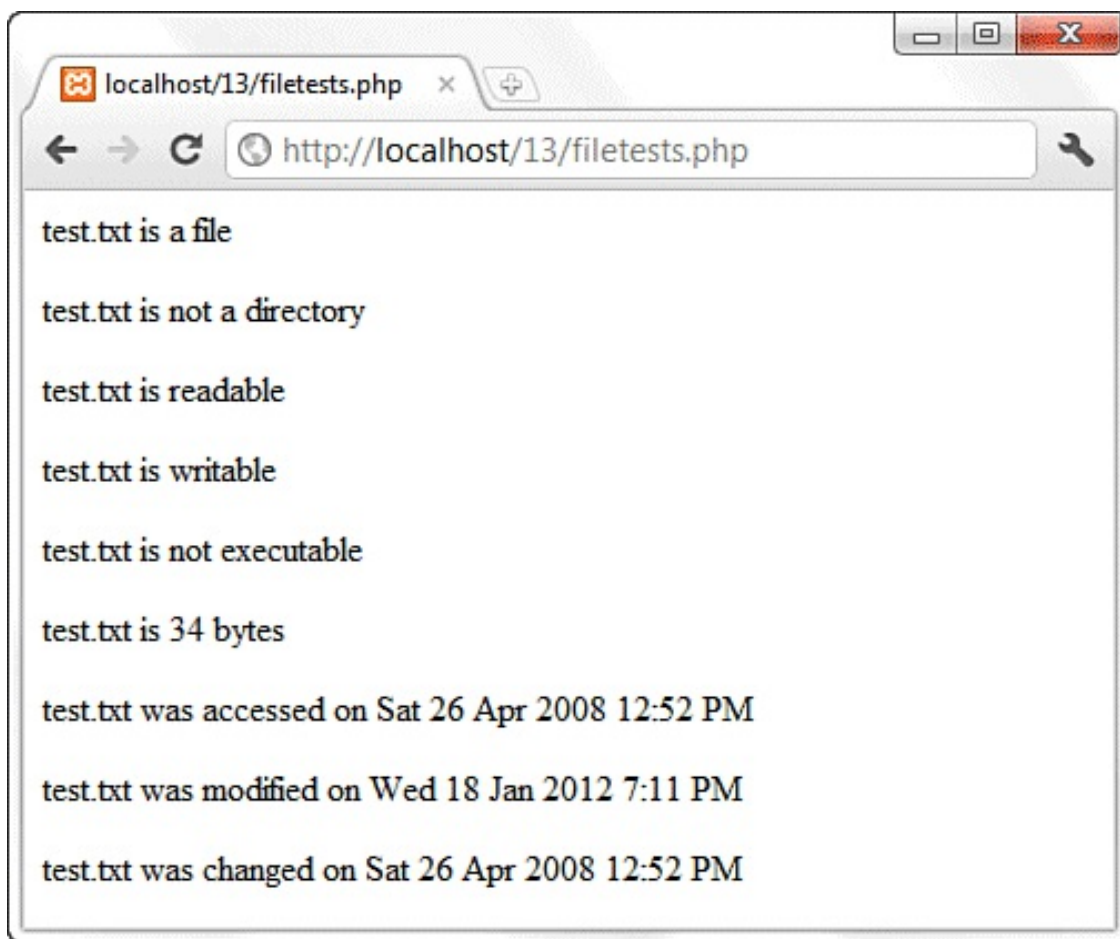


图13-2 filetests.php的输出

注意，在进行某些测试的时候我们使用三元操作符作为一种紧凑的方式。让我们更详细地看看这样一个测试，如程序清单13.7的第7行所示。

```
echo "<p>$f is ".(is_file($f) ? "" : "not ")."a file</p>";
```

我们使用 `is_file()` 函数作为三元操作符的左表达式。如果它返回 `true`，将会返回一个空字符串，否则，将返回字符串“not”。三元表达式的返回值使用连接操作符添加到要显示的字符串。这条语句可以通过扩展变得更清楚些，如下所示。

```
$is_it = is_file($f) ? "" : "not ";  
echo "<p>".$f." is ".$is_it." a file</p>";
```

当然，我们可以设置用一条if语句使其变得更清楚，不过想象一下，如果使用如下的形式，函数将会变得多长。

```
if (is_file($f)) {  
    echo "<p>$f is a file</p>";  
} else {  
    echo "<p>$f is not a file</p>";  
}
```

由于这3种方法的结果是一样的，选择哪种方法由个人偏好而定。

## 13.3 创建并删除文件

如果一个文件还不存在，我们可以使用`touch()`函数来创建它。给定一个表示文件路径的字符串，`touch()`函数试图使用该名字创建一个空文件。如果这个文件已经存在，其内容不会被改变，但是修改日期将会更新，以反映出该函数的执行时间。

```
touch('myfile.txt');
```

我们可以使用`unlink()`函数来删除一个已有的文件，就像`touch()`函数一样，`unlink()`也接受一个文件路径。

```
unlink('myfile.txt');
```

在UNIX系统上创建、删除、读取、写入和修改文件的所有函数都需要设置正确的文件或目录许可。

## 13.4 打开一个文件供写入、读取或添加

在我们开始操作一个文件之前，必须先打开它以便读取或写入，或者这两种任务都执行。PHP 提供了 `fopen()` 函数来做到这一点，并且这个函数需要一个包含了文件路径的字符串，后面跟着一个字符串，其中包含了文件的打开模式。最常见的模式是读取（`r`）、写入（`w`）和添加（`a`）。

`fopen()` 函数返回一个文件源，我们稍后可以使用它来操作被打开的文件。要打开一个文件以供读取，我们使用如下的方式。

```
$fp = fopen("test.txt", "r");
```

使用如下方式来打开一个文件以便写入。

```
$fp = fopen("test.txt", "w");
```

要打开一个文件以便添加内容，即在文件末尾添加数据，我们使用，如下方式。

```
$fp = fopen("test.txt", "a");
```

如果由于许多原因而不能打开文件，`fopen()` 函数返回 `false`。因此，在使用文件之前测试函数的返回值，是个不错的方法。我们可以使用如下一条 `if` 语句来做到这一点。

```
if ($fp = fopen("test.txt", "w")) {  
    // do something with the $fp resource  
}
```

或者，如果一个基本文件无法打开，我们可以使用一个逻辑操作符

来结束执行，示例如下。

```
($fp = fopen("test.txt", "w")) or die("Couldn't open file, sorry");
```

如果fopen()函数返回true，表达式的剩下部分将不会解析，即die()函数（该函数会向浏览器写入一条消息并结束脚本）不会执行。否则，or操作符的右侧表达式将会解析，即调用die()函数。

假设这一切都正常，并且我们继续使用打开的文件，当我们完成的时候应该记得关闭它。我们可以通过调用fclose()函数来做到这一点，这需要从一次成功的fopen()调用所返回的文件源作为参数。

```
fclose($fp);
```

这个曾经可用的文件源（\$fp）现在对你已经不可用了。



## 13.5 读取文件

PHP提供了几个函数来从文件读取数据。这些函数使得我们能够按照字节读取、按照整行读取，甚至是按照单个字符读取。

### 13.5.1 使用fgets()和feof()从一个文件读取行

当我们打开一个文件以供读取的时候，你可能想一行一行地访问它。要从一个打开的文件读取行，我们可以使用fgets()函数，这需要从fopen()返回的文件源作为其参数。我们还必须传递一个整数作为fgets()的第二个参数，它指明了如果函数没有遇到一个行末或文件末尾而应该读取的字节数。fgets()函数读取文件，直到它遇到一个换行字符（“\n”），或者达到长度参数中指定的字节数，或者是到达文件的末尾，不管它先遇到哪一个，示例如下。

```
$line = fgets($fp, 1024); // where $fp is the file resource returned by fopen()
```

尽管可以使用 fgets()来读取行，但是我们需要某种方法来告知何时到达文件的末尾。feof()函数可以做到这一点，当到达了文件的末尾的时候，该函数返回true，否则返回false。feof()函数需要一个文件源作为参数，示例如下。

```
feof($fp); // where $fp is the file resource returned by fopen()
```

现在，我们有足够的信息来一行一行地读取一个文件，如程序清单13.8所示。

程序清单13.8 打开一个文件并一行一行地读取

```
1: <?php
2: $filename = "test.txt";
3: $fp = fopen($filename, "r") or die("Couldn't open $filename");
4: while (!feof($fp)) {
5:     $line = fgets($fp, 1024);
6:     echo $line."<br/>";
7: }
8: ?>
```

把上述代码作为readlines.php保存到Web服务器的文档根目录下，并且在Web浏览器中运行它，输出将会如图13-3所示（你的示例文本文件的内容可能会有所不同）。

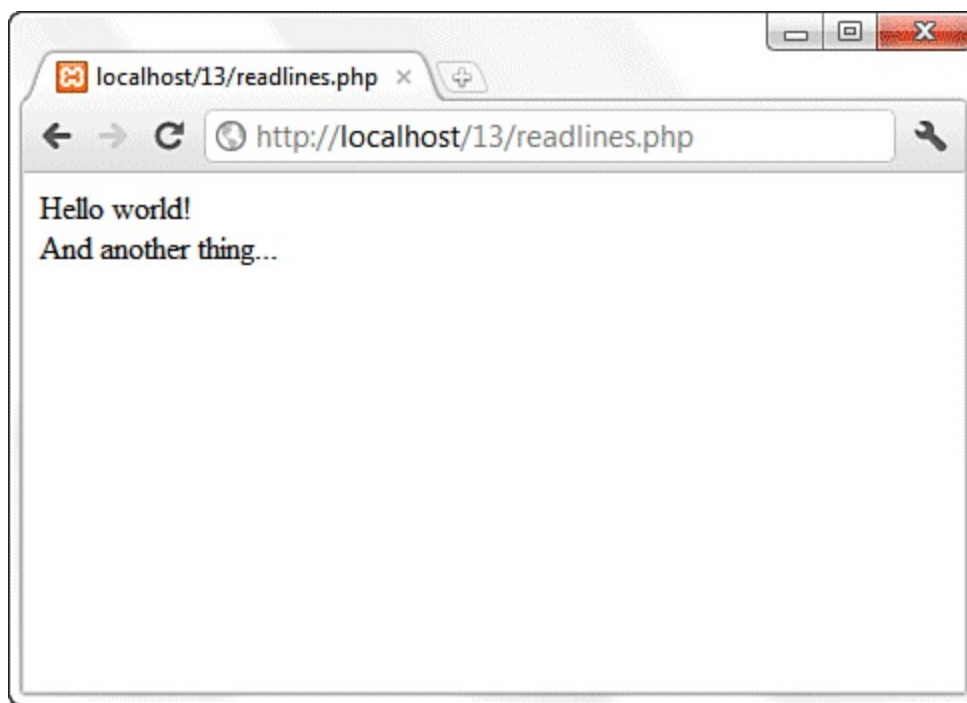


图13-3 readlines.php的输出

我们在第3行调用fopen()函数，使用想要读取的文件的名字作为参数。如果没有读取文件，我们使用or die()结构来确保结束脚本的执行。如果文件不存在或文件的许可不允许访问这个文件，通常会发生这种情况。

文件内容的实际读取发生在第4行的 `while` 语句。`while`语句的测试表达式每次迭代都调用`feof()`函数，当它返回 `true` 的时候循环结束。换句话说，循环持续直到到达文件末尾。在代码块内部，我们在第5行使用`fgets()`从文件提取一行（或者 1024字节，不管这一行够不够 1024 字节）。我们把结果赋给`$line` 并且在第6行将其显示到浏览器，为了增强可读性，在最后附加一个`<br/>`标记。

### 13.5.2 使用`fread()`函数从文件读取任意数量的数据

我们可以选择按照任意定义的大小来读取一个文件，而不是按照行来读取文本。`fread()`函数接受一个文件源以及想要读取的字节数作为参数。`fread()`函数返回我们所请求的那么多的数据，除非到达文件的末尾。

```
$chunk = fread($fp, 16);
```

程序清单 13.9 修改了前面的例子，以便以8字节的数据块来读取数据，而不是按照行来读取。

如果把这段代码作为`readlines2.php`文件保存到Web服务器的文档根目录下，并且通过Web浏览器来运行它，将会看到如图13-4所示的结果。

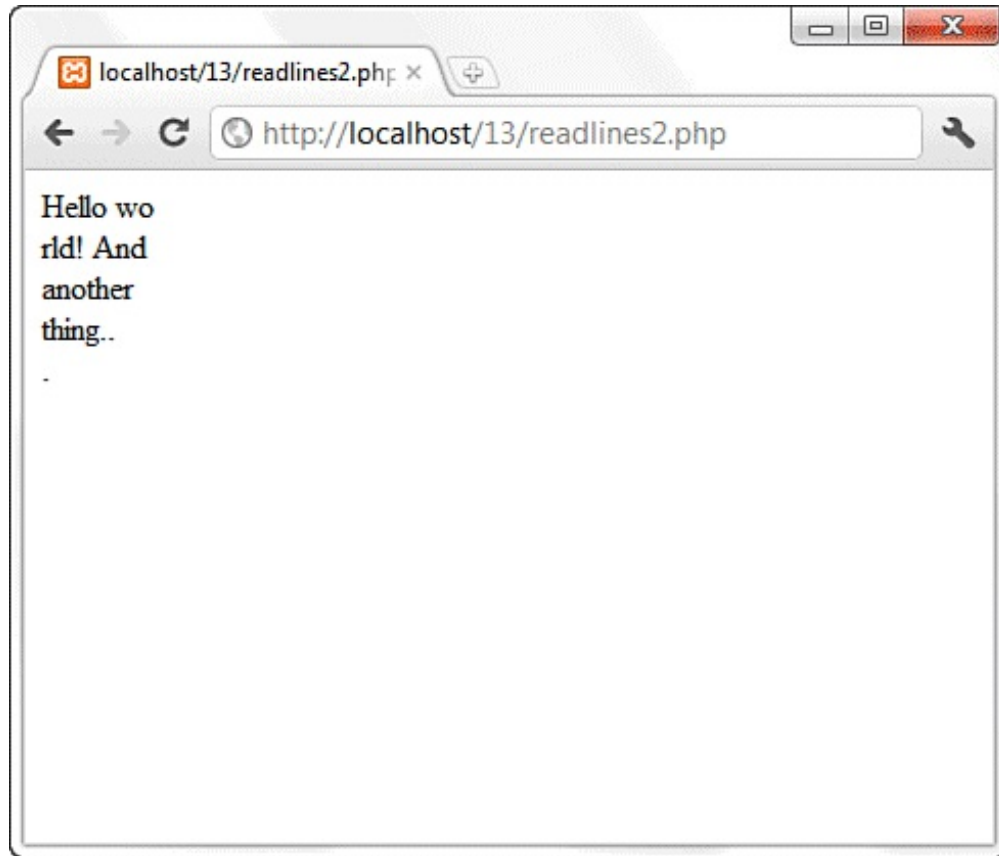


图13-4 readlines2.php的输出

程序清单13.9 使用fread()读取文件

---

```
1: <?php
2: $filename = "test.txt";
3: $fp = fopen($filename, "r") or die("Couldn't open $filename");
4: while (!feof($fp)) {
5:     $chunk = fread($fp, 8);
6:     echo $chunk."<br/>";
7: }
8: ?>
```

---

尽管fread()函数允许你定义从一个文件获取的数据的量，但是，它不允许你决定从哪个位置开始获取。我们可以使用fseek()函数来手动地设置它。

fseek()函数允许我们在一个文件中修改当前位置。它需要一个文件

源以及一个整数作为参数，这个整数表示从文件的开始跳过的偏移量（字节数）。

```
fseek($fp, 64);
```

程序清单13.10使用fseek()和fread()来把一个文件后半部分输出到浏览器。

程序清单13.10 使用fseek()来移动文件

---

```
1: <?php
2: $filename = "test.txt";
3: $fp = fopen($filename, "r") or die("Couldn't open $filename");
4: $fsize = filesize($filename);
5: $halfway = (int)($fsize / 2);
6: echo "Halfway point: ".$halfway." <br/>\n";
7: fseek($fp, $halfway);
8: $chunk = fread($fp, ($fsize - $halfway));
9: echo $chunk;
10: ?>
```

---

如果把上述代码作为readseek.php保存到Web服务器的文档根目录下，并且通过Web浏览器来运行它，将会看到如图13-5所示的结果。

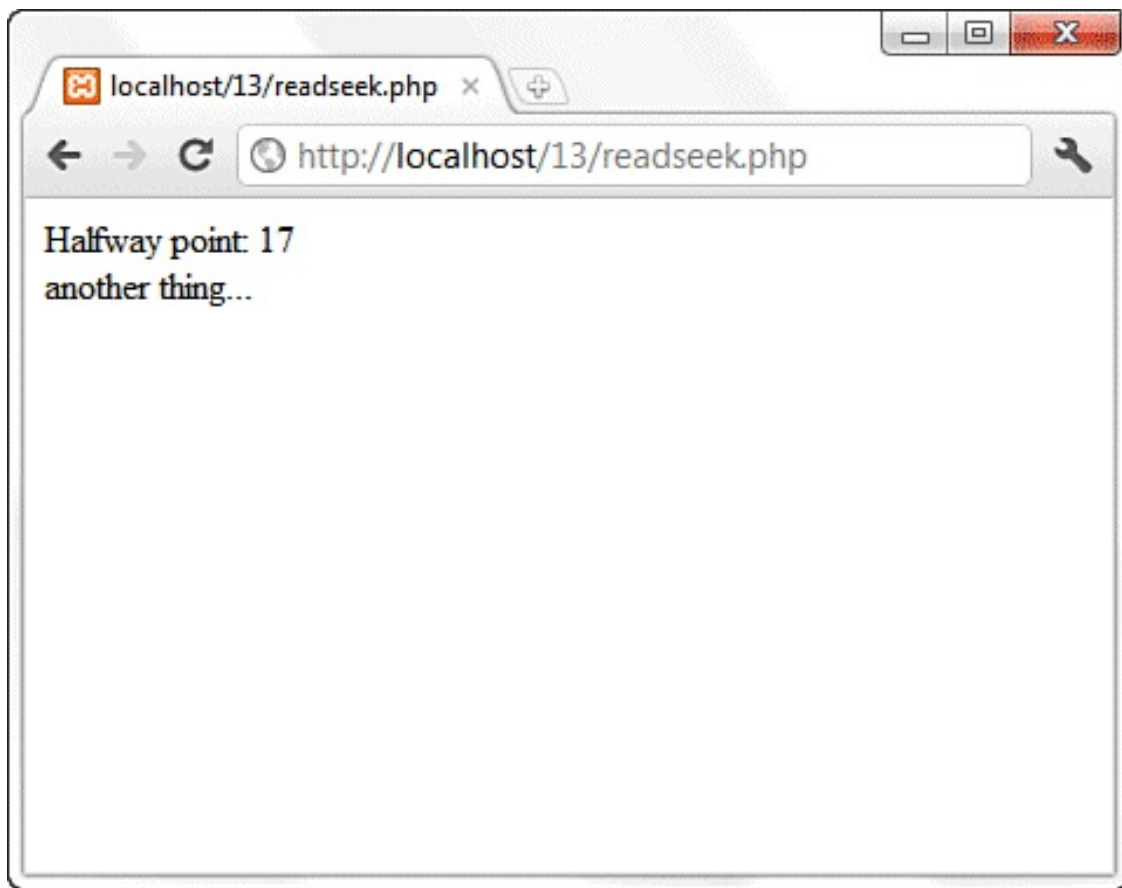


图13-5 readseek.php 的输出

我们在第5行通过把`filesize()`的返回值除以2，计算出文件的一半的位置。在第7行，把这个位置用作`fseek()`函数的第2个参数，跳到这个文本文件一半的位置。最后，在第8行调用`fread()`函数来提取文件的另一半，并且将结果输出到显示器。

### 13.5.3 使用**fgetc()**从文件读取字符

`fgetc()`函数和 `fgets()`函数很相似，只不过每次调用它的时候只从文件返回一个字符。由于一个字符总是一个字节的大小，`fgetc()`不需要长度参数。我们只需传递给它一个文件源，示例如下。

```
$char = fgetc($fp);
```

程序清单 13.11 创建了一个循环，它每次从 test.txt 文件读出一个字符，把每个字符作为单独一行输出到浏览器中。

程序清单13.11 使用fgetc()来移动文件

---

```
1: <?php
2: $filename = "test.txt";
3: $fp = fopen($filename, "r") or die("Couldn't open $filename");
4: while (!feof($fp)) {
5:     $char = fgetc($fp);
6:     echo $char."<br/>";
7: }
8: ?>
```

---

如果把上述代码作为readchars.php文件保存到Web服务器的文档根目录下，并且通过Web浏览器来运行它，将会看到如图13-6所示的结果。

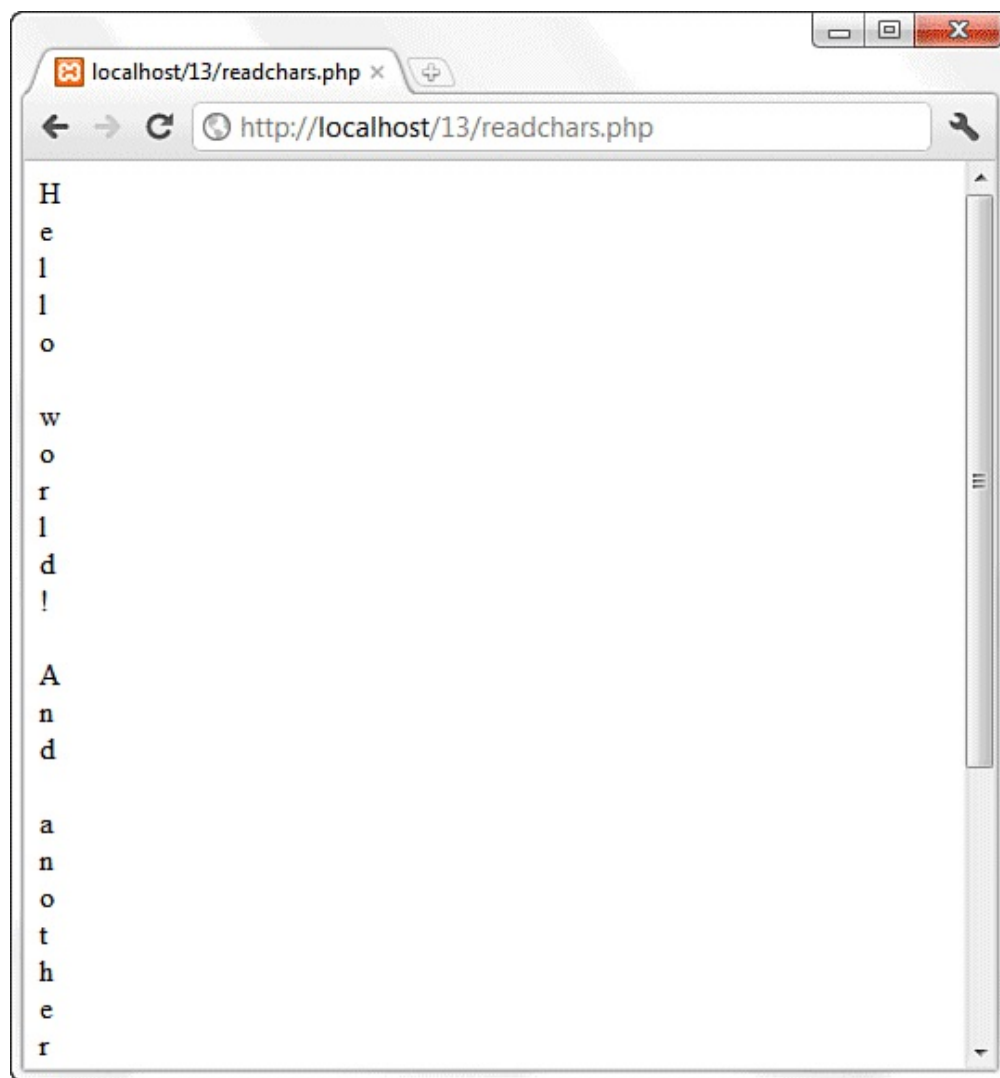


图13-6 readchars.php的输出

#### 13.5.4 用file\_get\_contents()读取文件内容

现在已经掌握了使用fopen()来创建一个文件资源并且对该资源进行一项或多项操作，可以通过使用file\_get\_contents()将整个文件读入到一个字符串中，从而来做各种操作。例如，如下的单行代码，将名为test.txt的文件的内容读入到一个名为\$content的变量中。

```
$content = file_get_contents("test.txt");
```



这个函数还提供了一些其他的可能性，下面给出了调用 `file_get_contents()` 时候可用的选项。

```
file_get_contents ($filename [$use_include_path [, $context [, $offset [,  
    $maxlen ]]]]);
```

这些选项描述如下。

- **use\_include\_path** ——这是一个布尔值，它表明函数是否应该搜索文件名的完整 `include_path`。
- **context** ——这是 `stream_context_create()` 创建的一个上下文资源；如果不需要的话，使用 `NULL`。
- **offset** ——读取从哪里开始，例如，5表示文件的第5个字符。注意，可以对非本地文件（例如URL）使用 `file_get_contents()`，在这种情况下，不能使用 `offset`。
- **maxlen** ——读取数据的最大长度，以字节为单位。默认情况下读取所有的数据。

如果前面函数的精细度控制在你的脚本中不是必须的，那么，`file_get_contents()` 也完全满足很多基本的文件读取目的。

## 13.6 写入文件或向文件添加内容

写入文件与向文件添加内容的过程是相同的，不同之处仅在于它们调用`fopen()`函数的模式。当我们写入一个文件的时候，调用`fopen()`函数的时候可以使用模式参数“w”，示例如下。

```
$fp = fopen("test.txt", "w");
```

所有后续的写入都发生在文件的开头处。如果这个文件还不存在，就创建它。如果这个文件已经存在，之前的所有内容都销毁并且由你写入的数据替代。

当我们添加到一个文件时，在`fopen()`调用中使用模式参数“a”，示例如下。

```
$fp = fopen("test.txt", "a");
```

所有后续的写入文件的内容都将添加到现有内容的末尾。如果你试图向一个不存在的文件添加内容，会先创建这个文件。

### 13.6.1 使用**fwrite()**或**fputs()**写入文件

`fwrite()`函数接受一个文件源和一个字符串，然后把字符串写入到文件中。`fputs()`函数也以相同的方式工作，示例如下。

```
fwrite($fp, "hello world");  
fputs($fp, "hello world");
```

写入到文件相当简单。程序清单13.12使用**fwrite()**写入到文件。然后，我们使用**fputs()**把另一个字符串添加到同一个文件。

---

```
1: <?php
2: $filename = "test.txt";
3: echo "<p>Writing to ".$filename." ... </p>";
4: $fp = fopen($filename, "w") or die("Couldn't open $filename");
5: fwrite($fp, "Hello world\n");
6: fclose($fp);
7: echo "<p>Appending to ".$filename." ...</p>";
8: $fp = fopen($filename, "a") or die("Couldn't open $filename");
9: fputs($fp, "And another thing\n");
10: fclose($fp);
11: ?>
```

---

当我们从Web浏览器运行这个脚本，屏幕的输出如下。

```
Writing to test.txt ...
Appending to test.txt ...
```

如果打开test.txt文件或者使用readlines.php来读取其内容，你将会发现文件现在包含如下内容。

```
Hello world
And another thing
```

### 13.6.2 使用file\_put\_contents()写文件内容

和前面介绍的file\_get\_contents()函数很相似，file\_put\_contents()函数以一种更为流线化的方式来进行文件操作。特别的，file\_put\_contents()直接模拟了依次调用fopen()、fwrite()和fclose()，来将数据写入文件。

下面展示了调用file\_put\_contents()的时候允许使用的参数。

```
file_put_contents ($filename, $data [, $flags [, $context]]);
```

这些额外的选项说明如下。

- **data** ——包含了要写的数据的一个字符串或数组。

- **flags** ——一个或多个FILE\_USE\_INCLUDE\_PATH（是否在include\_path中查找目标文件）、FILE\_APPEND（如果文件已经存在的话，是否将数据添加到一个文件后面）和LOCK\_EX（是否在写入目标文件时获取一个专用的锁），这些选项用|操作符组合起来。
- **context** ——stream\_context\_create()创建的一个环境资源；如果不需要的话，使用NULL。

如果使用file\_put\_contents()重新编写程序清单13.12，它看上去如程序清单13.13所示。

程序清单13.13 写入并添加到一个文件

---

```
1: <?php
2: $filename = "test.txt";
3: echo "<p>Writing to ".$filename." ... </p>";
4: file_put_contents ($filename, "Hello world\n");
5: echo "<p>Appending to ".$filename." ...</p>";
6: file_put_contents ($filename, "And another thing\n", FILE_APPEND);
7: ?>
```

---

### 13.6.3 使用flock()锁定文件

如果只有一个用户访问脚本的话，刚刚学习的读取和补充文件的方法会工作得很好。然而，在现实使用中，我们期望更多的用户可以访问站点以及其中的脚本，所以或多或少会有同时的访问。假设有两个用户要同时执行写入到一个文件的脚本，这个文件会很快面目全非。

PHP提供了flock()函数来避免这种问题。flock()函数锁定一个文件，以警告其他的进程，在当前进程使用这个文件的时候，不要再写入或读取这个文件。flock()函数需要一个来自打开的文件的有效文件源以

及一个整数，这个整数表示你想要设置的锁定的种类。PHP为我们可能需要的每个整数都提供了预定义的常量。表13-1列出了我们可以应用于一个文件的3种锁定。

表13-1 flock()函数的整数参数

常量	整数	锁定	类型说明
LOCK_SH	1	共享	允许其他进程读取文件但是阻止写入（在读取一个文件的时候使用）
LOCK_EX	2	独占	阻止其他进程读取文件或者写入文件（在写入一个文件时使用）
LOCK_UN	3	释放	释放一个共享锁定或独占锁定

我们应该在调用fopen()之后直接调用flock()，然后，在关闭一个文件之前再次调用它来释放锁定。如果锁定没有释放，将不能够读取或写入文件。下面是这个事件序列的一个例子。

```
$fp = fopen("test.txt", "a") or die("Couldn't open file.");
flock($fp, LOCK_EX); // create exclusive lock
// write to the file
flock($fp, LOCK_UN); // release the lock
fclose($fp);
```

提示：

你知道吗？要了解关于文件锁定的更多信息，请参见位于<http://www.php.net/flocks> 的PHP

手册中的flock()函数条目。

## 13.7 使用目录

既然可以创建、测试和写入文件，那么让我们把注意力转向目录。PHP提供了很多用来使用目录的函数。让我们看一下如何创建目录、删除目录和从中读取内容。

### 13.7.1 使用`mkdir()`创建目录

`mkdir()`函数使得我们能够创建一个目录。`mkdir()`函数需要一个字符串和一个八进制整数，字符串表示到我们想要创建的目录的路径，整数表示我们想要为目录设置的模式。别忘了，我们使用0开头来指定一个八进制数（以8为基数），例如0777或0400。

模式参数只在UNIX系统下有效。这个模式应该由从0到7之间的3个数字组成，它们分别表示对目录所有者、组和任何人的许可权限。如果成功地创建了目录，`mkdir()`函数返回true，否则返回false。如果`mkdir()`失败，通常是因为包含目录不允许带有脚本的用户ID的进程写入。

如果我们不知道如何正确地设置UNIX目录许可，如下例子中的一个应该符合你的需求。除非真的需要将自己的目录设置为完全可写的，否则你可能应该使用0755，它使得目录完全可读但是无法向其中写入，除了目录的所有者之外。

```
mkdir("testdir", 0777); // global read/write/execute permissions
mkdir("testdir", 0755); // world/group: read/execute; owner: read/write/execute
```

### 13.7.2 使用`rmdir()`删除一个目录

如果运行脚本的进程有权利这么做，并且目录为空的话，`rmdir()`函数允许我们从文件系统删除一个目录。`rmdir()`函数只需要一个字符串作为参数，它表示需要删除的目录的路径。

```
rmdir("testdir");
```

### 13.7.3 使用`opendir()`打开一个目录以供读取

在阅读一个目录的内容之前，必须首先获得一个目录源。我们可以使用`opendir()`函数来做到这一点。`opendir()`函数需要一个字符串，它表示要打开的目录的路径，`opendir()`函数返回一个目录句柄，除非这个目录不存在或者不可读。如果目录不存在或者不可读，它返回`false`，如下所示。

```
$dh = opendir("testdir");
```

在这个例子中，`$dh`就是打开目录的目录句柄。

### 13.7.4 使用`readdir()`从一个目录读取内容

就像使用`fgets()`函数从一个文件中读取一行一样，我们可以使用`readdir()`从一个目录读取一个文件或目录名。`readdir()`函数需要一个目录句柄并且返回包含项目名的一个字符串。如果到达了目录的末尾，`readdir()`返回`false`。注意，`readdir()`只是返回其项目的名字，而不是完整的路径。程序清单13.14显示了一个目录的内容。

程序清单13.14 使用`readdir()`列出一个目录的内容



---

```
1: <?php
2: $dirname = ".";
3 : $dh = opendir($dirname) or die("Couldn't open directory");
4:
5: while (!(($file = readdir($dh)) === false ) ) {
6:     if (is_dir("$dirname/$file")) {
7:         echo "(D) ";
8:     }
9:     echo $file."<br/>";
10: }
11: closedir($dh);
12: ?>
```

---

把上述代码作为readdir.php保存到Web服务器的文档根目录下，并且通过Web浏览器来运行它，其输出如图13-7所示。

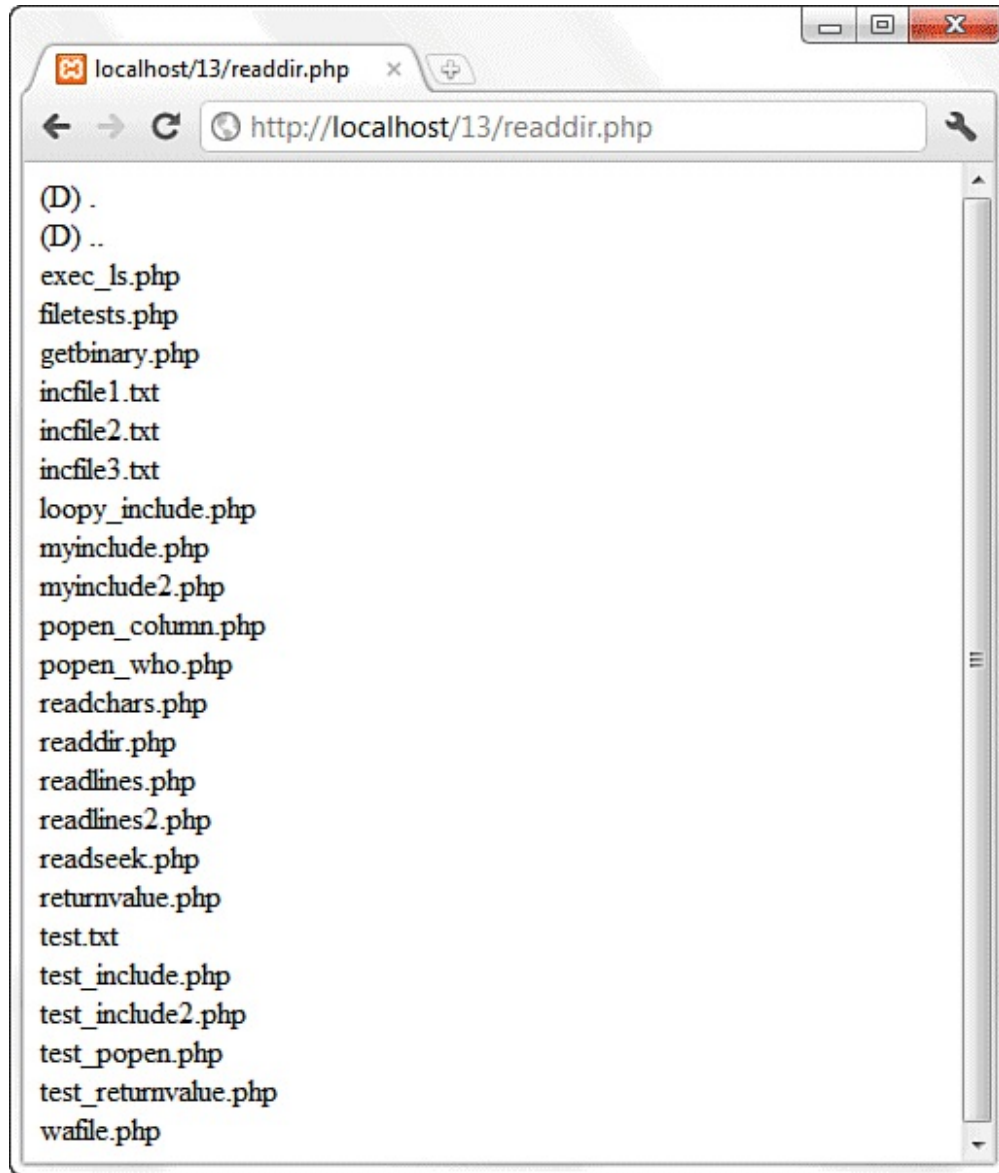


图13-7 readdir.php的输出

我们在第3行使用`opendir()`函数打开目录以便读取，并且从第5行开始，使用一条`while`语句来遍历其中的每个元素。我们调用`readdir()`语句作为`while`语句的测试表达式的一部分，并且将其结果赋给`$file`变量。

在 `while` 语句体的内部，我们使用`$dirname` 变量和`$file` 变量一起来创建一个完整的文件路径，然后，在第6行测试这个路径。如果这个路

径表示一个目录，我们让第7行在浏览器上显示（D）。最后，在第9行显示文件名或目录名。

程序清单 13.14 在while语句的测试中使用了一个谨慎的结构。大多数PHP程序员（包括我自己在内）都愿意使用如下所示的代码。

```
while ($file = readdir($dh)) {  
    echo $file."<br/>";  
}
```

在这个例子中，测试readdir()的返回值，并且，由于除“0”以外的任何字符串都求值为true，这里不会有什么问题。然而，假设一个目录包含4个文件：0、1、2和3。在我自己的系统上，前面的代码输出如下。

```
.  
..
```

当循环到达名为0的文件的时候，readdir()返回的字符串求值为false，这会导致循环结束。程序清单13.14中的方法使用===来检查，看readdir()的返回值确实不为false。在测试中，只有结果0才会得到false，因此，我们避免了问题。

**提示：**

如果你发现目录列表中的项目的顺序是随意的，可能是因为由文件系统自己决定顺序。如果想要让项目按照指定的方式排列，必须把内容读取到一个数组中，然后，按照想要的方式排序它们并顺序地显示。

## 13.8 使用popen()打开到进程和离开进程的管道

在本章前面，我们学习了如何使用fopen()函数打开一个文件以供读取和写入。现在，我们将会看到，可以使用popen()函数来打开到一个进程的管道。

popen()函数可以如下使用。

```
$file_pointer = popen("some command", mode)
```

模式可以是r（读取）或w（写入）。

程序清单 13.15 设计用来产生一个错误，它试图打开一个文件以供读取，而这个文件并不存在。

程序清单13.15 使用popen()来读取一个文件

---

```
1: <?php
2: $file_handle = popen("/path/to/fakefile 2>&1", "r");
3: $read = fread($file_handle, 2096);
4: echo $read;
5: pclose($file_handle);
6: ?>
```

---

我们首先在第2行调用popen()函数，试图打开一个文件以供读取。在第3行，读取存储在\$file\_handle指针中的任何错误消息，并在第4行显示到屏幕上。最后，第5行关闭了在第2行中打开的文件指针。如果把这段代码保存为test\_popen.php，将其放置到文档根目录下，并且通过Web浏览器访问它，你将会看到如下所示的错误消息。

```
The system cannot find the path specified.
```

程序清单13.16也使用了popen()来读取一个进程的输出，在这个例子中，是UNIX的who命令的输出。

程序清单13.16 使用popen()读取UNIX who命令的输出（只适用于UNIX）

---

```
1: <?php
2: $handle = popen("who", "r");
3: while (!feof($handle)) {
4:     $line = fgets($handle,1024);
5:     if (strlen($line) >= 1) {
6:         echo $line."<br/>";
7:     }
8: }
9: pclose($handle);
10: ?>
```

---

在第2行，当为了读取文件而使用popen()的时候，返回了一个文件指针。第3行开始了一个while循环，它会从进程读取输出的每一行，并且最终在代码的第6行显示每一行输出。如果这些行中包含有信息的话，连接在第9行关闭。

如果把这段代码保存为popen\_who.php，并将其放置到文档根目录下，当使用Web浏览器访问它们时，你将会看到如下所示的内容（当然，这和你实际信息而不是我的实际信息相关）。

```
julie pts/0 Mar 14 06:19 (adsl-63-206-120-158.dsl.snfc21.pacbell.net)
```

程序清单13.17展示了如何以写入模式来使用popen()，从而把数据传递给外部应用程序。这个例子中的外部应用程序叫做column。这段脚本的目标是获取多维数组的一个元素，并且将其以表格的形式输出到一个ASCII文件中。

程序清单13.17 使用popen()把数据传递到UNIX column命令（只适用于UNIX）

---

```
1: <?php
2: $products = array(
3:     array("HAL 2000", 2, "red"),
4:     array("Tricorder", 3, "blue"),
5:     array("ORAC AI", 1, "pink"),
6:     array("Sonic Screwdriver", 1, "orange")
7: );
8:
9: $handle = popen("column -tc 3 -s / > /somepath/purchases.txt", "w");
10: foreach ($products as $p) {
11:     fputs($handle, join('/', $p)."\n");
12: }
13: pclose($handle);
14: echo "done";
15: ?>
```

---

在第2行到第7行，我们定义了一个名为\$products的多维数组，并且在其中放置了4个条目表示带有名称、数量和颜色的4种产品。在第9行，使用popen()以写入格式来向column应用程序发送一条命令。这条命令向column应用程序发送参数，让它使用或作为字段分隔符，把输入格式化为一个具有3列的表格。输出将会发送到一个名为purchases.txt的文件中，请确保把路径修改为你的系统上存在的一个路径。

在第10行到第12行，我们使用foreach来遍历\$products数组并且把每个元素发送给打开文件指针。然后，用join()函数把数组转换为字符串，带有附加于其上的指定的分隔符。我们在第13行关闭了打开文件指针，并且在第14行把一条状态消息显示到屏幕上。

如果我们把这段代码保存为popen\_column.php，将其放置到文档根目录下，并且使用Web浏览器来访问它，应该在指定的位置中创建一个文件。查看在我自己机器上所创建的文件，可以看到如下文本。

```
HAL 2000          2  red
Tricorder         3  blue
ORAC AI           1  pink
Sonic Screwdriver 1  orange
```

你的系统中可能有也可能没有 `column` 程序，但是本节说明了打开一条到应用程序的管道的逻辑和语法。请自行采用可用的其他程序来尝试这些逻辑。

## 13.9 使用exec()运行命令

exec()函数是我们可以用来向 shell 传送命令的几个函数之一。exec()函数需要一个字符串并且可选地接受一个数组变量和一个标量变量，字符串表示要运行的命令的路径，数组变量将会包含命令的输出，标量变量将包含返回值（1或0），示例如下。

```
exec("/path/to/somecommand", $output_array, $return_val);
```

程序清单13.18使用exec()函数和基于 shell的ls命令来产生一个目录列表。

程序清单13.18 使用exec()函数和ls生成一个目录列表（只适用于UNIX）

---

```
1: <?php
2: exec("ls -al .", $output_array, $return_val);
3: echo "Returned ".$return_val."<br/><pre>";
4: foreach ($output_array as $o) {
5:     echo $o."\n";
6: }
7: echo "</pre>";
8: ?>
```

---

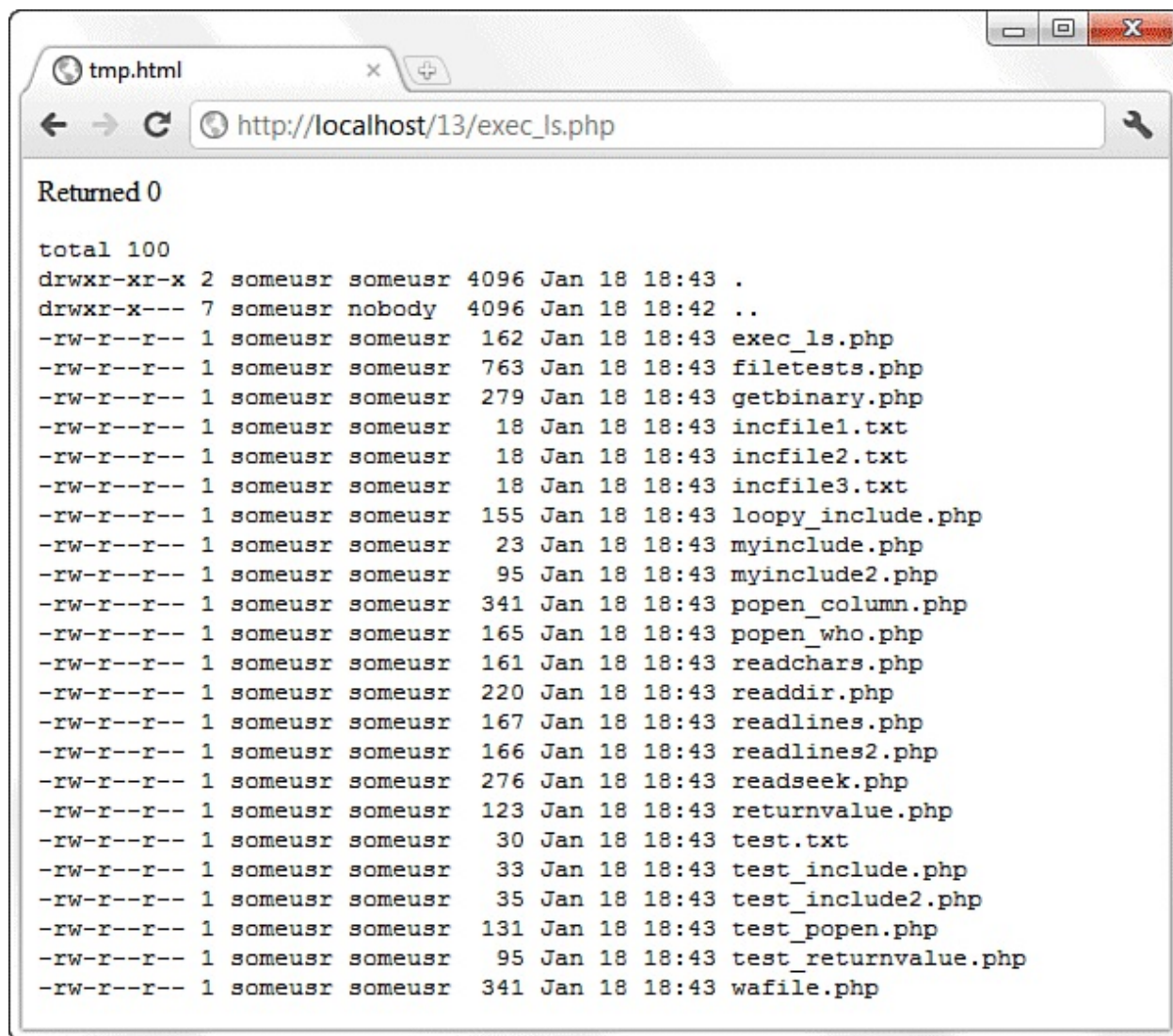
在第 2 行，我们使用 exec()函数来执行 ls 命令。我们把命令的输出放入到\$output\_array数组，并且把返回值放入到变量\$return\_val。第3行只是显示了返回值，而第4行到第6行的 foreach循环显示出\$output\_array中的每个元素。

### 提示：

第3行中的字符串包含了一个开始的<pre>标记，第 7行提供了一个结束标记。这只是通过使用HTML预处理的文本来确保目录列表可读。



如果把上述代码保存为exec\_ls.php，将其放置到文档根目录中，并且使用Web浏览器访问它，将会看到如图13-8所示的结果。当然，这和你的实际信息而不是我的实际信息相关。



```
Returned 0

total 100
drwxr-xr-x  2 someusr someusr 4096 Jan 18 18:43 .
drwxr-x---  7 someusr nobody 4096 Jan 18 18:42 ..
-rw-r--r--  1 someusr someusr 162 Jan 18 18:43 exec_ls.php
-rw-r--r--  1 someusr someusr 763 Jan 18 18:43 filetests.php
-rw-r--r--  1 someusr someusr 279 Jan 18 18:43 getbinary.php
-rw-r--r--  1 someusr someusr  18 Jan 18 18:43 incfile1.txt
-rw-r--r--  1 someusr someusr  18 Jan 18 18:43 incfile2.txt
-rw-r--r--  1 someusr someusr  18 Jan 18 18:43 incfile3.txt
-rw-r--r--  1 someusr someusr 155 Jan 18 18:43 loopy_include.php
-rw-r--r--  1 someusr someusr  23 Jan 18 18:43 myinclude.php
-rw-r--r--  1 someusr someusr  95 Jan 18 18:43 myinclude2.php
-rw-r--r--  1 someusr someusr 341 Jan 18 18:43 popen_column.php
-rw-r--r--  1 someusr someusr 165 Jan 18 18:43 popen_who.php
-rw-r--r--  1 someusr someusr 161 Jan 18 18:43 readchars.php
-rw-r--r--  1 someusr someusr 220 Jan 18 18:43 readdir.php
-rw-r--r--  1 someusr someusr 167 Jan 18 18:43 readlines.php
-rw-r--r--  1 someusr someusr 166 Jan 18 18:43 readlines2.php
-rw-r--r--  1 someusr someusr 276 Jan 18 18:43 readseek.php
-rw-r--r--  1 someusr someusr 123 Jan 18 18:43 returnvalue.php
-rw-r--r--  1 someusr someusr  30 Jan 18 18:43 test.txt
-rw-r--r--  1 someusr someusr  33 Jan 18 18:43 test_include.php
-rw-r--r--  1 someusr someusr  35 Jan 18 18:43 test_include2.php
-rw-r--r--  1 someusr someusr 131 Jan 18 18:43 test_popen.php
-rw-r--r--  1 someusr someusr  95 Jan 18 18:43 test_returnvalue.php
-rw-r--r--  1 someusr someusr 341 Jan 18 18:43 wafire.php
```

图13-8 exec\_ls.php的输出

尽管PHP精彩纷呈，但还是有这样的时候，即我们需要把某种功能整合到基于PHP的应用程序中，但是其他人已经用Perl编写了做同样事情的代码。在这种情况下，没有必要再重新发明轮子，因为我们可以直

接使用`exec()`来访问已有的Perl脚本并利用其功能。然而，别忘了，调用一个外部应用程序总是会增加脚本的负担，无论从时间上还是从内存使用上都是如此。

## 13.10 使用system()或passthru()运行命令

system()函数和exec()函数类似，因为它们都启动一个外部应用程序，并且它使用一个标量变量来存储返回值，示例如下。

```
system("/path/to/somecommand", $return_val);
```

system()函数和exec()函数的不同在于，它直接向浏览器输出信息，而没有编程的干预。如下的代码段使用system()向 man 命令显示一个 man 页面，这个页面使用<pre></pre>标记对格式化。

```
<?php
echo "<pre>";
system("man man | col -b", $return_val);
echo "</pre>";
?>
```

类似地，passthru()函数的语法和system()函数相同，但行为不同。使用passthru()的时候，shell命令的任何输出在返回给你的过程中不会缓存，这适用于运行产生二进制数据而不是简单的文本数据的命令。一个例子就是使用shell工具来定位一个图像并且将其发送回浏览器，如程序清单13.19所示。

程序清单13.19 使用passthru()输出二进制数据

---

```
1: <?php
2: if ((isset($_GET['imagename'])) && (file_exists($_GET['imagename']))) {
3:     header("Content-type: image/gif");
4:     passthru("giftopnm ".$_GET['imagename']." |
5:         pnmscale -xscale .5 -yscale .5 | ppmtojpeg");
6: } else {
7:     echo "The image ".$_GET['imagename']." could not be found";
8: }
9: ?>
```

---

**提示：**

这个脚本中使用的shell工具，包括giftopnm、pnmscale和ppmtogif，也可能在你的系统上没有安装。如果没有安装，可能要从你的操作系统的发布 CD 上安装它们，但是不用担心，这只是一个例子。这里只是通过这个程序清单来理解使用passthru()函数的概念。

假设把这个文件命名为getbinary.php，它将会像下面这样从HTML调用。

```
">
```

在程序清单13.19的第2行，测试用户输入以确保涉及到的文件（根据HTML代码段，是文件test.gif）是存在的。由于这个脚本将会向浏览器输出GIF数据，相应的标头在第3行设置。

在第4行和第5行，passthru()函数连续执行3个不同的命令，giftopnm、pnmscale和ppmtogif，它们把图像缩放为原始高度和原始宽度的50%。passthru()函数的输出，也就是新的图像数据被发送到浏览器。

**提示：**

在这个及其他和系统相关的例子中，我们可以使用escapeshellcmd()或escapeshellarg()函数在用户输入中转义元素。这么做就确保了用户不会诱使系统执行随意的命令，例如删除重要系统文件或者重新设置密码。这些函数提供了用户输入的第二个实例，示例如下。

```
$new_input = escapeshellcmd($_GET['someinput']);
```

然后，我们可以在脚本中的其他地方引用\$new\_input变量，而不是\$\_GET['someinput']。使用这两条命令，加上确保编写的脚本使其只执行我们希望它执行的任务，而不是执行来自用户的命令；以上就是确保系统安全的一种方法。

## 13.11 小结

在本章中，我们学习了如何使用include等语句（include\_once()、require()和作为额外增添的require\_once()）来把文件包含到文档中，并且执行包含文件中的任何PHP代码。学习了如何使用某些PHP文件测试函数，以及用来按行、按字符或者按任意数据量读取文件的函数。学习了如何写入到文件，不管是替代已有内容还是添加到已有内容的后面，并且我们学习了如何创建、移动和读取目录。

我们还学习了和系统及其外部应用程序通信的各种方法。尽管PHP是快速而健壮的语言，我们还是会发现直接利用已有的其他语言（如C或Perl）的脚本会更节约成本和时间。我们可以使用popen()、exec()、system()和passthru()函数访问这些外部应用程序。

我们学习了如何使用popen()把数据导出给一条命令，对于从标准输入接受数据的应用程序以及当我们想要解析由应用程序发送给我们的数据的时候，这非常有用。我们还学习了使用exec()和system()把命令传递给shell以及获取用户输入。我们还学习了使用passthru()函数来接受作为一条shell命令的结果的二进制数据。

既然可以更多地使用文件系统，我们就可以保存和访问更多数量的数据。然而，如果我们需要从大的文件中查找数据，这样的脚本会变得相当慢。当发生这种情况的时候，我们需要使用一个数据库系统，稍后将介绍它。

## 13.12 Q&A

**Q:** `include`语句会使脚本变慢吗？

**A:** 由于一个被包含的文件必须由引擎打开和解析，这增加了负担。然而，相对降低的性能负载，可重用代码库带来的好处更大。

**Q:** 如果一个文件无法打开以便写入或读取，我们总是应该结束脚本执行吗？

**A:** 应该总是考虑到这种可能性。如果脚本绝对依靠我们想要使用的文件，你可能希望使用`die()`函数，向浏览器显示一条含有信息的错误消息。在重要程度较低的情况下，我们仍然需要考虑到失效，可以将其写入到一个日志文件中。

**Q:** 在网上哪里可以获得关于安全的更多信息？

**A:** 对Web安全的一个权威性介绍是Lincoln Stein所编写的文档《The World Wide Web Security FAQ》，可以在<http://www.w3.org/Security/Faq/> 找到。

## 13.13 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 使用什么函数把库代码添加到当前运行的脚本中？
2. 使用什么函数来确认一个文件是否在你的文件系统上？
3. 如何确定一个文件的大小？
4. 使用什么函数来打开一个文件以供读取和写入？
5. 使用什么函数来从一个文件读取一行数据？
6. 如何获知到达一个文件的末尾？
7. 使用什么函数来把一行数据写入到一个文件？
8. 如何打开一个目录以供读取？
9. 在我们打开一个目录之后，使用什么函数来读取一个目录项的名字？
10. 使用哪个函数来打开一个通向进程的管道？
11. 当我们打开一个连接之后，如何从一个进程读取数据？写入数据呢？
12. 在把用户输入传递到一个shell命令之前，如何转义用户输入使其更安全一些？



## 解答

1. 我们可以使用`require()`或`include()`语句来把PHP文件加入到当前的文档中，也可以使用`include_once()`或`require_once()`。
2. 我们可以使用`file_exists()`函数来测试一个文件的存在性。
3. `filesize()`函数返回一个文件的字节大小。
4. `fopen()`函数打开一个文件。它接受一个文件的路径以及一个表示模式的字符。它返回一个文件源。
5. `fgets()`函数读取数据直到达到我们传递给它的缓冲大小、或者到达行的末尾，或者到达文档的末尾，不管上述哪种情况，以先出现的为准。
6. 当传递给`feof()`函数的文件源到达文件末尾的时候，`feof()`函数返回`true`。
7. 我们可以使用`fputs()`函数来把数据写入一个文件。
8. `opendir()`函数允许我们打开一个目录以供读取。
9. `readdir()`函数返回一个打开的目录中的一个目录项的名字。
10. `popen()`函数用来打开到一个进程的管道。
11. 我们可以从一个进程读取数据或把数据写入一个进程，就好像

我们使用一个打开的文件一样，即使用`feof()`和`fgets()`来读取，使用`fputs()`来写入。

12. 如果用户输入是你的 `shell`命令的一部分，可以使用`escapeshellcmd()`或`escapeshellarg()`函数来适当地转义他们。

## 思考题

1. 创建一个表单，它接受一个用户的第一个名字和第二个名字。  
创建一个脚本把这些数据保存到一个文件。
2. 创建一个脚本，读取我们在思考题 1 中创建的数据文件。除了将其内容显示到浏览器（在每行添加一个<br/>标记），还显示出包含文件中的行数和文件大小的摘要。

## 第14章 使用图像

在本章，我们将学到如下内容：

- 如何修改**PHP**来增加和图形相关的功能。
- 如何创建一个新的图像。
- 如何修改已有的图像。

**PHP**的标准安装拥有很多内建的函数，可以用来动态地创建和操作图像。常见用法包括图表的创建，以及修改已有的图像来显示水印。只需要略作调整，我们就可以更多地扩展这些函数。在本章中，我们学习使用**PHP**函数创建和操作图像的基础。

## 14.1 理解图像创建过程

使用PHP创建一个图像，不像使用一个绘图程序创建图像那样（例如，Sumo Paint、Corel DRAW或Windows Draw）。这里没有指向、点击或拖拽颜料桶到一个预先定义的空间以填充图像等操作。同样，这里也没有另存为功能，使绘图程序自动创建一个GIF、JPEG、PNG等，仅仅因为我们要求它这么做。

相反，我们必须变成一个绘图应用程序。作为程序员，我们必须告诉PHP引擎在绘图过程中的每一步做什么。我们负责使用单个的PHP函数来定义颜色、绘制和填充形状、确定和调整图像大小，并且把图像保存为一个指定的文件类型。然而，如果你理解这个过程中的每一步，并且按照顺序完成任务的话，这并不像看上去那么难。

### 注意：

本章中的例子并不是PHP创建图像最令人激动的例子，但是，受篇幅限制，本书正文中只能介绍这些内容。我们可以保证，对于使用PHP中与图像相关的函数，这些例子将会给你打下很好的基础。

## 关于颜色

当在图像相关的脚本中定义颜色的时候，我们将使用 RGB 颜色系统。针对红色、绿色和蓝色（R、G 和 B）中的每一项，使用从 0 到 255 之间的十进制数值，这样就可以定义一个具体的颜色了。0表示没有任何数量的该颜色，而255表示该颜色的最大量。

例如，纯红色的RGB值是（255， 0， 0），或者说完全分配红色值，而没有绿色和蓝色。类似的，纯绿色的值为（0， 255， 0），纯蓝色的值为（0， 0， 255）。另一方面，白色的RGB值为（255， 255， 255），而黑色的RGB值为（0， 0， 0）。漂亮的暗紫色的RGB值是（153， 51， 153），而浅灰色的RGB值是（204， 204， 204）。为了方便参考，可以查看[http://en.wikipedia.org/wiki/Web\\_colors](http://en.wikipedia.org/wiki/Web_colors) 的颜色和RGB值的列表。

## 14.2 对PHP的必要修改

PHP的当前发布版本包含了Thomas Boutell的GD图形库的一个绑定版本。包含了这个库也就不再需要下载和安装第三方的库了，但是这个库需要在安装的时候激活。

要在安装的时候使得GD库可用，Linux/UNIX用户必须在准备编译PHP的时候向配置参数添加如下内容。

```
--with-gd
```

再次运行PHP配置程序之后，你必须像第4章所介绍的那样使用一次make和make install的过程。想要使用GD的Windows用户只需要激活php\_gd2.dll作为php.ini文件中的一个扩展，正如我们在第4章所介绍的。

当使用GD库的时候，除非我们安装了其他的库，否则被限制只能使用GIF格式的文件。

使用GIF文件能够很好地满足我们的需要，但是，如果我们想要在Linux/UNIX上创建JPEG或PNG文件，就需要下载并安装一些其他库，并且对我们的PHP安装做一些修改（Windows平台上的安装中包含了这些库并且可用）。

- JPEG库及其信息可以在<http://www.ijg.org/> 找到。
- PNG库及其信息可以在<http://www.libpng.org/pub/png/libpng.html> 找到。
- 如果想要使用PNG文件，还应该安装zlib库，可以

在<http://www.zlib.net/> 找到。

按照这些站点上的说明来安装这些库。在安装之后，Linux/UNIX用户必须通过把如下的内容添加到PHP配置参数中，从而再次配置和重新编译PHP（假设我们想要使用所有3个库，如果不是的话，只需要添加需要使用的库）。

```
--with-jpeg-dir=[path to jpeg directory]  
--with-png-dir=[path to PNG directory]  
--with-zlib=[path to zlib directory]
```

再次运行PHP配置程序之后，你必须像第4章所介绍的那样使用一次 `make` 和 `make install` 的过程。这样，我们的库就激活了并准备好使用了。



## 14.3 绘制一个新的图像

用来创建一个新图像的基本的PHP函数叫做ImageCreate(), 但是, 创建一个图像并不像调用函数那么简单。创建一个图像是一个逐步的过程, 并且包括几种不同的PHP函数的使用。

创建一个图像从ImageCreate()函数开始, 但是, 这个函数所作的只是为新图像留出一块画布区域。下面的代码创建了一个300像素宽、300像素高的画布区域。

```
$myImage = ImageCreate(300,300);
```

现在定义好了一块画布, 接下来可以定义一些颜色供新图像使用。如下的例子使用ImageColorAllocate()函数和RGB值定义了5种这样的颜色(分别是黑色、白色、红色、绿色和蓝色)。

```
$black = ImageColorAllocate($myImage, 0, 0, 0);  
$white = ImageColorAllocate($myImage, 255, 255, 255);  
$red = ImageColorAllocate($myImage, 255, 0, 0);  
$green = ImageColorAllocate($myImage, 0, 255, 0);  
$blue = ImageColorAllocate($myImage, 0, 0, 255);
```

### 提示:

在你的脚本中, 所分配的第一种颜色用作图像的背景颜色。在这个例子中, 背景颜色是黑色(你也可以根据使用环境, 将\$background\_coloror变量命名为其他有意义的名字)。

既然我们了解了设置绘图画布和分配颜色的基础知识, 现在可以继续学习绘制形状并且真正地把图像输出到一个Web浏览器了。

### 14.3.1 绘制形状和线条

有如下几个PHP函数可以帮助我们在画布上绘制形状和线条。

- `ImageEllipse()`用来绘制一个椭圆。
- `ImageArc()`用来绘制一个部分椭圆。
- `ImagePolygon()`用来绘制一个多边形。
- `ImageRectangle()`用来绘制一个矩形。
- `ImageLine()`用来绘制一个线条。

使用这些函数需要先提前考虑一下，因为我们必须设置所进行的绘制从哪一点开始到哪一点结束。这些函数中的每一个都使用x轴坐标和y轴坐标来表示在画布上绘制的起点。我们还必须定义要绘制到离x轴和y轴多远的位置。

例如，如下代码在画布上绘制了一个矩形，该矩形从点(15,15)开始，并且水平方向 80个像素，垂直方向140个像素的长度，因此，线条在点(95,155)结束。另外，这些线条将使用红色绘制，该颜色已经在变量\$red中定义了。

```
ImageRectangle($myImage, 15, 15, 95, 155, $red);
```

如果想要绘制大小相同但线条颜色为白色的另一个矩形，而它从一个矩形结束的点开始，可以使用如下的代码。

```
ImageRectangle($myImage, 95, 155, 175, 295, $white);
```

最后，为了添加与第一个矩形对齐的其他红色矩形，但是，从白色矩形的最右边的点开始，使用如下代码。

```
ImageRectangle($myImage, 175, 15, 255, 155, $red);
```

程序清单14.1给出了目前为止的一个图像创建脚本，其中添加了几

行代码用来把图像输出到Web浏览器。

程序清单14.1 创建一个新的图像

---

```
1: <?php
2: //create the canvas
3: $myImage = ImageCreate(300,300);
4:
5: //set up some colors for use on the canvas
6: $black = ImageColorAllocate($myImage, 0, 0, 0);
7: $white = ImageColorAllocate($myImage, 255, 255, 255);
8: $red = ImageColorAllocate($myImage, 255, 0, 0);
9: $green = ImageColorAllocate($myImage, 0, 255, 0);
10: $blue = ImageColorAllocate($myImage, 0, 0, 255);
11:
12: //draw some rectangles
13: ImageRectangle($myImage, 15, 15, 95, 155, $red);
14: ImageRectangle($myImage, 95, 155, 175, 295, $white);
15: ImageRectangle($myImage, 175, 15, 255, 155, $red);
16:
17: //output the image to the browser
18: header ("Content-type: image/png");
19: ImagePng($myImage);
20:
21: //clean up after yourself
22: ImageDestroy($myImage);
23: ?>
```

---

第18行到第19行把图像数据的流输出到 Web 浏览器，它首先使用所创建图像的MIME类型来发送相应的header()函数。然后，使用ImageGif()、ImageJpeg()或ImagePng()函数相应地输出数据流，这个例子输出的是一个PNG文件。在第22行，我们使用ImageDestroy()函数清空ImageCreate()函数在脚本开头所使用的内存。

把上述程序清单保存为magecreate.php并且将其放置在Web服务器的文档根目录下。访问它的时候，结果看上去如图14-1所示，只不过还带有颜色。

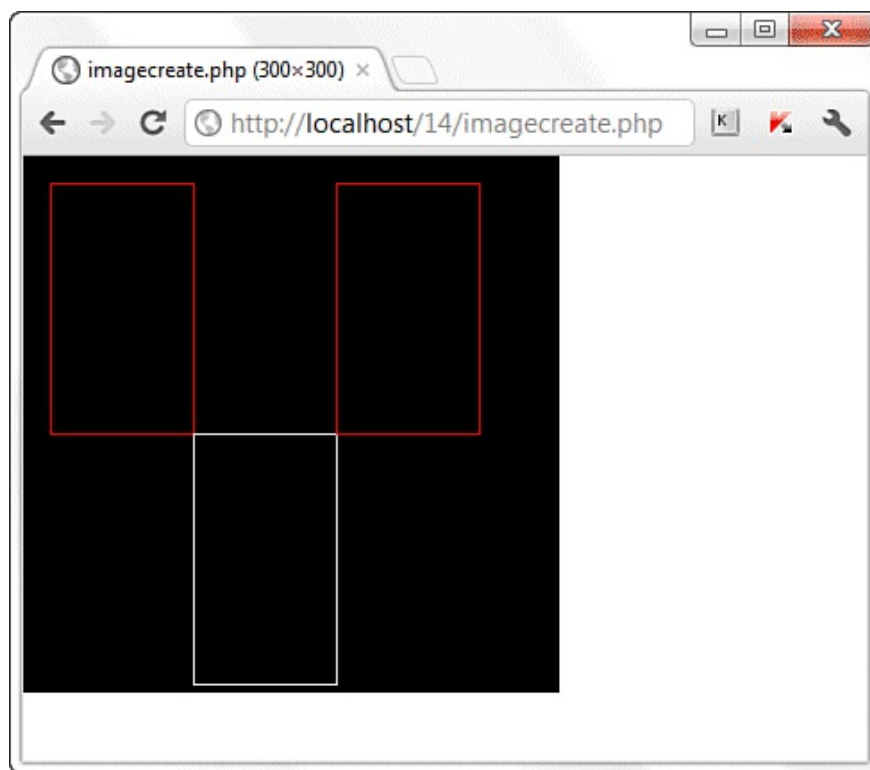


图14-1 带有三个绘制的矩形的画布

### 14.3.2 使用颜色填充

程序清单 14.1 的输出只是生成了矩形的边框。PHP 还有如下专门的图像函数，是设计用来填充区域的。

- `ImageFilledEllipse()`用来填充一个椭圆。
- `ImageFilledArc()`用来填充一个部分椭圆。
- `ImageFilledPolygon()`用来填充一个多边形。
- `ImageFilledRectangle()`用来填充一个矩形。

这些函数的用法和它们对应的非填充函数的用法是相同的。在程序清单14.2中，把非填充函数替换为设计用来填充一个区域的函数。换句话说，只有第13行和第15行代码修改了。

---

```
1: <?php
2: //create the canvas
3: $myImage = ImageCreate(300,300);
4:
5: //set up some colors for use on the canvas
6: $black = ImageColorAllocate($myImage, 0, 0, 0);
7: $white = ImageColorAllocate($myImage, 255, 255, 255);
8: $red = ImageColorAllocate($myImage, 255, 0, 0);
9: $green = ImageColorAllocate($myImage, 0, 255, 0);
10: $blue = ImageColorAllocate($myImage, 0, 0, 255);
11:
12: //draw some rectangles
13: ImageFilledRectangle($myImage, 15, 15, 95, 155, $red);
14: ImageFilledRectangle($myImage, 95, 155, 175, 295, $white);
15: ImageFilledRectangle($myImage, 175, 15, 255, 155, $red);
16:
17: //output the image to the browser
18: header ("Content-type: image/png");
19: ImagePng($myImage);
20:
21: //clean up after yourself
22: ImageDestroy($myImage);
23: ?>
```

---

把上述程序清单保存为imagecreatefill.php并且将其放置到Web服务器的文档根目录下。访问它的时候，将会得到如图14-2所示的结果，也是带有颜色的。

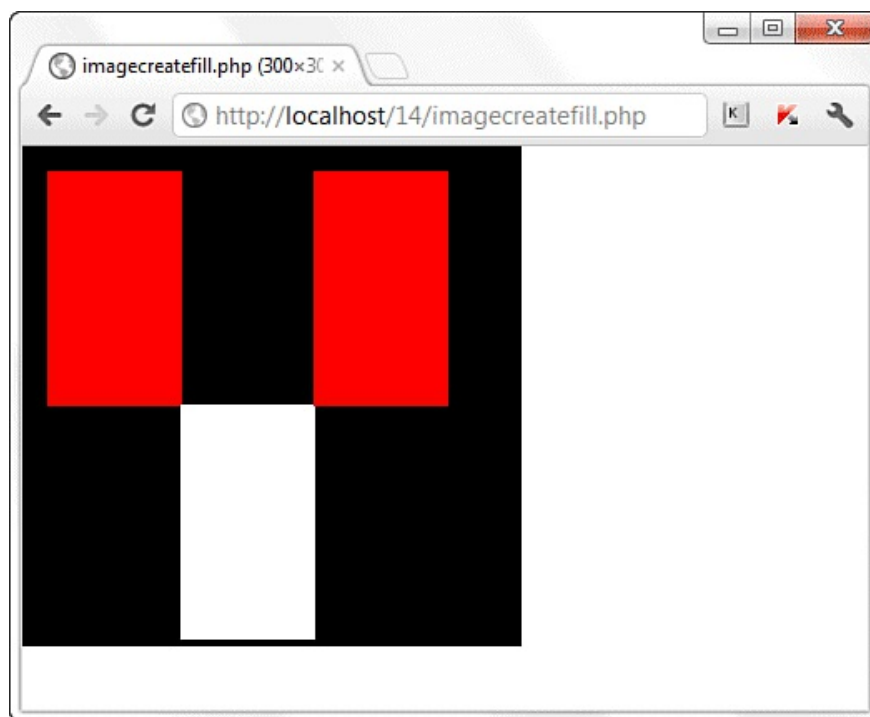


图14-2 带有三个绘制并填充的矩形的画布

## 14.4 绘制有趣的饼图

前面的例子有点枯燥，但是它们向你介绍了创建图像的过程，包括定义画布、定义颜色，然后绘制和填充。我们可以按照同样的事件顺序来扩展脚本，针对数据点使用静态或动态的数据从而创建图表。程序清单 14.3 绘制了一个基本的饼图。第 1 行到第 10 行看上去和前面的程序清单完全相同，因为它们只是设置画布的大小和所用的颜色。

程序清单14.3 一个基本的饼图

---

```
1:  <?php
2:  //create the canvas
3:  $myImage = ImageCreate(300,300);
4:
5:  //set up some colors for use on the canvas
6:  $white = ImageColorAllocate($myImage, 255, 255, 255);
7:  $red   = ImageColorAllocate($myImage, 255, 0, 0);
8:  $green = ImageColorAllocate($myImage, 0, 255, 0);
9:  $blue  = ImageColorAllocate($myImage, 0, 0, 255);
10:
11:  //draw a pie
12:  ImageFilledArc($myImage, 100, 100, 200, 150, 0, 90, $red, IMG_ARC_PIE);
13:  ImageFilledArc($myImage, 100, 100, 200, 150, 90, 180, $green,
    IMG_ARC_PIE);
14:  ImageFilledArc($myImage, 100, 100, 200, 150, 180, 360, $blue,
    IMG_ARC_PIE);
15:
16:  //output the image to the browser
17:  header ("Content-type: image/png");
18:  ImagePng($myImage);
19:
20:  //clean up after yourself
21:  ImageDestroy($myImage);
22:  ?>
```

---

由于第一个定义的颜色是白色，画布的颜色将会是白色的。

在第12行到第14行，我们使用了ImageFilledArc()函数，它有如下几

个属性。

- 图像的标识符。
- 部分椭圆的中心点x坐标。
- 部分椭圆的中心点y坐标。
- 部分椭圆的宽度。
- 部分椭圆的高度。
- 部分椭圆的开始点。
- 部分椭圆的结束点。
- 颜色。
- 样式。

程序清单14.3的第14行绘制了一个弧形。

这个弧形应该使用定义的颜色\$blue 填充，并且应该使用 IMG\_ARC\_PIE 样式。IMG\_ARC\_PIE样式是用来显示的几种内建样式之一，这个样式表示创建一个圆角的边。

提示：

我们可以在<http://www.php.net/image> 的PHP手册中了解所有的不同样式。

把上述程序清单保存为 `imagecreatepie.php`，并且将其放置到 Web 服务器的文档根目录下。当访问它的时候，应该看到如图14-3所示的结果，不过是带有颜色的。



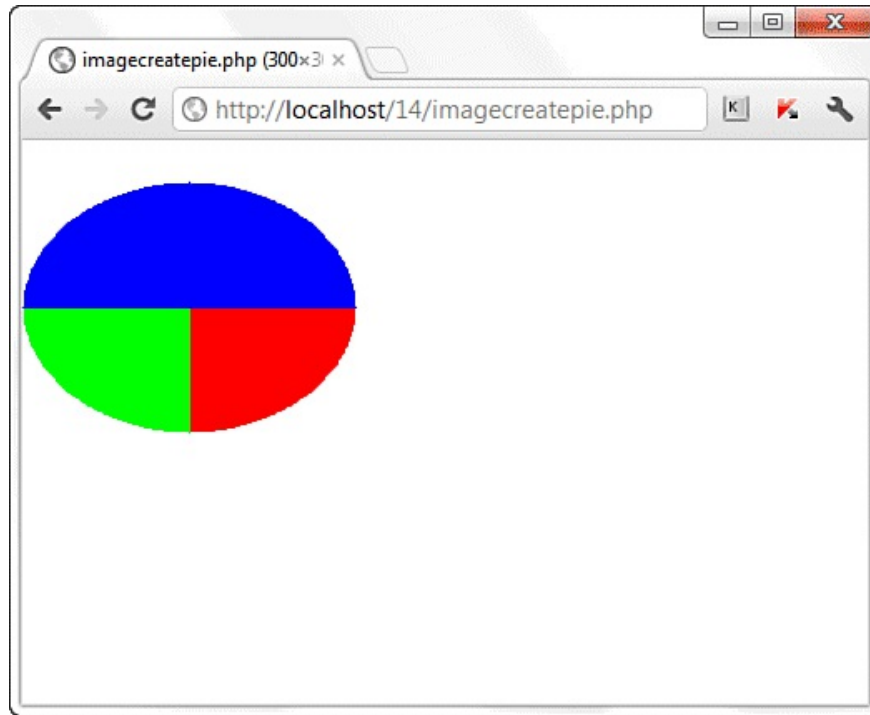


图14-3 带有分块的一个简单饼图

可以扩展程序清单14.3中的代码并得到一个3D外观的饼图。为了实现这一点，为边定义另外3种颜色。这些颜色可以比基本颜色深或浅，只要它们能够提供某种对比。下面的例子定义了3种较浅的颜色。

```
$lt_red = ImageColorAllocate($myImage, 255, 150, 150);  
$lt_green = ImageColorAllocate($myImage, 150, 255, 150);  
$lt_blue = ImageColorAllocate($myImage, 150, 150, 255);
```

要创建阴影效果，我们使用一个for循环从点（100，110）到（100，101）添加一系列较小的弧形，使用较浅的颜色作为填充颜色。

```
for ($i = 110;$i > 100;$i--) {  
    ImageFilledArc ($myImage, 100, $i, 200, 150, 0, 90, $lt_red, IMG_ARC_PIE);  
    ImageFilledArc ($myImage, 100, $i, 200, 150, 90, 180, $lt_green,  
        IMG_ARC_PIE);  
    ImageFilledArc ($myImage, 100, $i, 200, 150, 180, 360, $lt_blue,  
        IMG_ARC_PIE);  
}
```

程序清单14.4给出了绘制一个3D饼图的代码。

程序清单14.4 一个3D饼图

---

```
1: <?php
2: //create the canvas
3: $myImage = ImageCreate(300,300);
4:
5: //set up some colors for use on the canvas
6: $white = ImageColorAllocate($myImage, 255, 255, 255);
7: $red = ImageColorAllocate($myImage, 255, 0, 0);
8: $green = ImageColorAllocate($myImage, 0, 255, 0);
9: $blue = ImageColorAllocate($myImage, 0, 0, 255);
10: $lt_red = ImageColorAllocate($myImage, 255, 150, 150);
11: $lt_green = ImageColorAllocate($myImage, 150, 255, 150);
12: $lt_blue = ImageColorAllocate($myImage, 150, 150, 255);
13:
14: //draw the shaded area
15: for ($i = 110;$i > 100;$i--) {
16:     ImageFilledArc ($myImage,100,$i,200,150,0,90,$lt_red,IMG_ARC_PIE);
17:     ImageFilledArc ($myImage,100,$i,200,150,90,180,$lt_green,IMG_ARC_PIE);
18:     ImageFilledArc ($myImage,100,$i,200,150,180,360,$lt_blue,IMG_ARC_PIE);
19: }
20:
21: //draw a pie
22: ImageFilledArc($myImage, 100, 100, 200, 150, 0, 90, $red, IMG_ARC_PIE);
23: ImageFilledArc($myImage, 100, 100, 200, 150, 90, 180, $green, IMG_ARC_PIE);
24: ImageFilledArc($myImage, 100, 100, 200, 150, 180, 360, $blue, IMG_ARC_PIE);
25:
26: //output the image to the browser
27: header ("Content-type: image/png");
28: ImagePng($myImage);
29:
30: //clean up after yourself
31: ImageDestroy($myImage);
32: ?>
```

---

把上述程序清单保存为 `imagecreate3dpie.php`，并且将其放置到 Web 服务器的文档根目录下。当访问它的时候，应该看到如图14-4所示的结果，不过是带有颜色的。

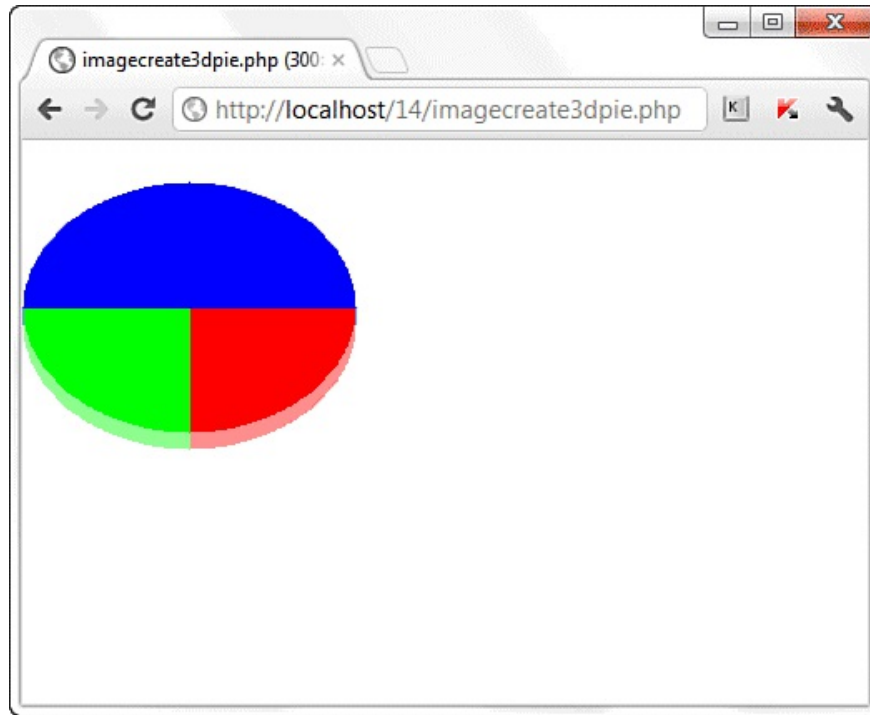


图14-4 一个带有分块的3D饼图

这只是展示某些图像绘制和填充函数的功能的基本例子。在下一节中，我们学习如何操作已有图像。

## 14.5 修改已有图像

从其他图像创建图像的过程，和创建一个新图像遵从同样的基本步骤，不同之处在于谁充当图像画布。在前面，我们使用ImageCreate()函数来创建一个新的画布。当从一个已有图像创建图像的时候，我们使用ImageCreateFrom\*()系列函数。

我们可以从已有的GIF、JPEG、PNG以及各种各样其他的图像类型来创建图像。从这些格式来创建图像的函数是ImageCreateFromGif()、ImageCreateFromJpg()和ImageCreateFromPng()等等。在下一个例子中，你可以看到从一个已有的图像创建一个新的图像是多么的容易。图14-5给出了基本图像。

程序清单14.5展示了如何使用已有的图像作为画布，然后在其上绘制一个椭圆。



图14-5 基本图像

程序清单14.5 从已有图像创建新图像

---

```
1: <?php
2: //use existing image as a canvas
3: $myImage = ImageCreateFromPng("baseimage.png");
4:
5: //allocate the color white
6: $white = ImageColorAllocate($myImage, 255, 255, 255);
7:
8: //draw on the new canvas
9: ImageFilledEllipse($myImage, 100, 70, 20, 20, $white);
10: ImageFilledEllipse($myImage, 175, 70, 20, 20, $white);
11: ImageFilledEllipse($myImage, 250, 70, 20, 20, $white);
12:
13: //output the image to the browser
14: header ("Content-type: image/png");
15: ImagePng($myImage);
16:
17: //clean up after yourself
18: ImageDestroy($myImage);
19: ?>
```

---

把上述程序清单保存为imagefrombase.php，并且将其放置到Web服务器的文档根目录下。当访问它的时候，将会看到如图14-6所示的结果。



图14-6 在已有图像上绘制图像

下一个例子把这个过程向前推进了几个步骤，并且使用了一些不同的图像修改函数。在这个例子中，已有的图像是4个PNG图像，每一个都是在一个灰色背景上有一个不同颜色的三角形分区。在程序清单 14.6 中，在每一个步骤中，我们将会把这些图像中的每一个分别叠加在另一个之上并将它们组合起来，以便灰色背景变成透明的，并且其下方的图像完全透过它显示出来。

在第3行，选择了一个图像作为基本图像。在这个例子中，它是 `img1.png`。第8行到第 11 行的 `for` 循环中处理了大量的工作。已经知道了你有 4 个图像并且第一个图像已经用作基本图像，那么，剩下的其他 3 个图像用来叠加并且变得透明。

程序清单14.6 叠加图像并且使它们变得透明

---

```
1: <?php
2: //select an image to start with
3: $baseimage = ImageCreateFromPng("img1.png");
4:
5: //loop through images #2 through the end
6: for($i=2; $i <5; $i++) {
7:     //allocate the transparent color, and stack
8:     $myImage = ImageCreateFromPng("img".$i.".png");
9:     $gray = ImageColorAllocate($myImage, 185, 185, 185);
10:    ImageColorTransparent($myImage, $gray);
11:    ImageCopyMerge($baseimage,$myImage,0,0,0,0,150,150,100);
12: }
13:
14: //output the image to the browser
15: header ("Content-type: image/png");
16: ImagePng($baseimage);
17:
18: //clean up after yourself
19: ImageDestroy($baseimage);
20: ?>
```

---

在第8行创建一个新层之后，其灰色区域表示为透明的，并且它被合并到基本图像的上面。当这个层叠加后，基本图像包含一个递增的图层编号，直到达到4层的总数。图像在第15行到第16行被发送到浏览器。

把上述程序清单保存为imagestacker.php并且将其放置到 Web 服务器的文档根目录下。当访问它的时候，将会看到如图14-7所示的结果。

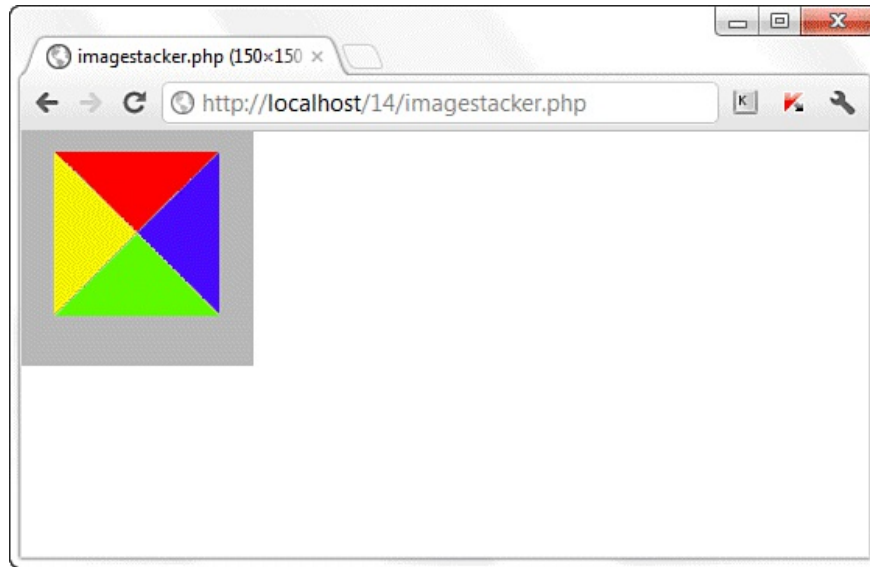


图14-7 叠加透明的图像来产生一个组合的图像



## 14.6 使用来自用户输入的图像创建图像

除了从其他图像创建图像以及绘制自己的图像之外，我们还可以根据用户输入来创建图像。这和如何创建脚本没有根本区别，只不过我们需要从一个表单收集值，而不是将值直接编码到脚本中。

在程序清单14.7中，我们创建了一个一体化的表单和脚本，它要求用户输入从图像大小到文本颜色以及背景颜色的各种属性，还要求输入一个消息字符串。我们将会学习`imagestring()`函数，它用来把一个字符串“写入”到一个图像上。

让我们来看看脚本，第 2 行到第 38 行表示用户输入表单，其他的代码行根据用户的要求处理所创建的图像。

程序清单14.7 从用户输入创建一个图像

```
1: <?php
2: if (!$_POST) {
3:     //show form
4:     ?>
5: <!DOCTYPE html>
6: <html>
7: <head>
8: <title>Image Creation Form</title>
9:
10: <style type="text/css">
11: fieldset{border: 0; padding: 0px 0px 12px 0px;}
12: fieldset label {margin-left: 24px;}
13: legend, label {font-weight:bold;}
14: </style>
15:
16: </head>
17: <body>
18: <h1>Create an Image</h1>
19: <form method="POST" action="<?php echo $_SERVER['PHP_SELF']; ?>">
20:
21: <fieldset>
22: <legend>Image Size:</legend><br/>
23: <label for="w">W:</label>
24: <input type="text" id="w" name="w" size="5" maxlength="5" />
25: <label for="h">H:</label>
26: <input type="text" id="h" name="h" size="5" maxlength="5" />
27: </fieldset>
28:
29: <fieldset>
30: <legend>Background Color:</legend><br/>
31: <label for="b_r">R:</label>
32: <input type="text" id="b_r" name="b_r" size="3" maxlength="3" />
33: <label for="b_g">G:</label>
34: <input type="text" id="b_g" name="b_g" size="3" maxlength="3" />
35: <label for="b_b">B:</label>
36: <input type="text" id="b_b" name="b_b" size="3" maxlength="3" />
37: </fieldset>
38:
39: <fieldset>
40: <legend>Text Color:</legend><br/>
41: <label for="t_r">R:</label>
42: <input type="text" id="t_r" name="t_r" size="3" maxlength="3" />
43: <label for="t_g">G:</label>
44: <input type="text" id="t_g" name="t_g" size="3" maxlength="3" />
45: <label for="t_b">B:</label>
46: <input type="text" id="t_b" name="t_b" size="3" maxlength="3" />
47: </fieldset>
48:
49: <p><label for="string">Text String:</label>
50: <input type="text" id="string" name="string" size="35" /></p>
51:
52: <p><label for="font_size">Font Size:</label>
53: <select id="font_size" name="font_size">
54: <option value="1">1</option>
55: <option value="2">2</option>
56: <option value="3">3</option>
57: <option value="4">4</option>
```

```
58: <option value="5">5</option>
59: </select></p>
60:
61: <fieldset>
62: <legend>Text Starting Position:</legend><br/>
63: <label for="x">X:</label>
64: <input type="text" id="x" name="x" size="3" maxlength="3" />
65: <label for="y">Y:</label>
66: <input type="text" id="y" name="y" size="3" maxlength="3" />
67: </fieldset>
68:
69: <button type="submit" name="submit" value="create">Create Image</button>
70: </form>
71: </body>
72: </html>
```

这段代码只是一个HTML表单，它定义了几个字段，用来获取图像的规格。可以看到，只有第1行到第4行是PHP代码，并且，这些代码行只查看表单是否已经提交了（如果\$\_POST超全局存在的话）。在查看之后，我们跳出了PHP，并且为表单提供了代码，只有表单还没有提交的时候，才会显示它；你可以将这段HTML放入到PHP代码中的一条echo语句中，但是，没有理由让PHP编译器来做这些工作。第24行到第26行中的表单字段定义了想要绘制的图像的宽度和高度。接下来，这段代码设置了用来为背景颜色（第32行、第34行和第36行）和文本颜色获取RGB值的几个字段（第42行、第44行和第46行）。

**提示：**

我们可以创建一个包含值0到255的下拉列表框，从而获取红色、绿色和蓝色的值。这将确保用户输入在要求的范围之内。

第50行包含了一个用来输入字符串的表单字段。这个字符串将会以指定的文本颜色绘制到图像的背景之上。第53行到第59行表示一个用

来选择字体大小的下拉列表。有5种大小，从1到5，分别表示默认的固定宽度字体。用来根据一个字符串绘制文本的函数，使用5种大小，从1到5表示（数值越高，字体越大），用于表示服务器上安装的默认的固定宽度字体。

提示：

我们可以使用 `imageloadfont()` 和 `imagefttext()` 函数来指定字体。详见<http://www.php.net/image>。

最后，第64行和第66行定义文本的开始位置。图像区域的左上角的X位置为0，Y位置也为0，向下增加10个像素的Y位置为10，向右增加10个像素的X位置为10，以此类推。

如果在这里终止了脚本并且结束了`if...else`语句和PHP代码块，当把这个脚本载入到Web浏览器的时候，我们将会看到如图14-8所示的一个表单。

The screenshot shows a web browser window with the title 'Image Creation Form' and the address bar displaying 'http://localhost/14/imagestring.php'. The main content area has a heading 'Create an Image' in a large, bold, serif font. Below the heading, there are several form fields: 'Image Size:' with 'W:' and 'H:' labels and text input boxes; 'Background Color:' with 'R:', 'G:', and 'B:' labels and text input boxes; 'Text Color:' with 'R:', 'G:', and 'B:' labels and text input boxes; 'Text String:' with a single-line text input box; 'Font Size:' with a dropdown menu showing '1'; 'Text Starting Position:' with 'X:' and 'Y:' labels and text input boxes. At the bottom left of the form is a button labeled 'Create Image'.

图14-8 用户输入表单来创建图像

只有在第 23 行以后，我们可以完成这个脚本并且产生带有文本字符串的图像，因此，看看程序清单 14.7 其余的内容。

程序清单14.7 从用户输入创建一个图像（续）

---

```
73: <?php
74: } else {
75:     //create image
76:     //create the canvas
77:     $myImage = ImageCreate($_POST['w'], $_POST['h']);
78:
79:     //set up some colors
80:     $background = ImageColorAllocate ($myImage, $_POST['b_r'],
81:         $_POST['b_g'], $_POST['b_b']);
82:     $text = ImageColorAllocate ($myImage, $_POST['t_r'],
83:         $_POST['t_g'], $_POST['t_b']);
84:
85:     // write the string at the top left
86:     ImageString($myImage, $_POST['font_size'], $_POST['x'],
87:         $_POST['y'], $_POST['string'], $text);
88:
89:     //output the image to the browser
90:     header ("Content-type: image/png");
91:     ImagePNG($myImage);
92:
93:     //clean up after yourself
94:     ImageDestroy($myImage);
95: }
96: ?>
```

---

第73行到第96行的主要部分我们之前已经见到过，只是这一次我们使用从超全局变量\$\_POST中提取的元素来取代直接编码的值。在第77行，我们使用来自表单的宽度值和高度值设置了初始图像。第80行到第83行定义了两种颜色变量，\$background和\$text，它们使用表单所提供的相应的RGB值。

**提示：**

在脚本中，这些颜色将不会给予实际的颜色名，因为我们不知道用户输入将会产生什么颜色，我们可能把这种颜色叫做\$red。但是如果用户将其定义为 0, 255, 0，我们就会傻眼了，因为这个 RGB 值表示绿色！因此，我们只是根据颜色的目的来命名颜色，而不是其外观。

第86行到第87行显示了这个脚本中的唯一的新项目，即

imagestring()函数的使用。这个函数的6个参数是图像流(\$myImage)、字体大小(\$\_POST[ 'font\_size' ])、开始点的X位置和Y位置(\$\_POST[ 'x ' ]和\$\_POST[ 'y ' ])、要绘制的字符串(\$\_POST[ 'string' ]), 以及绘制字符串的颜色(\$text)。第90行到第91行把图像输出到浏览器, 而且第94行销毁并清除图像创建过程。

如果我们把这个文件保存为imagecreate.php, 将其放置到Web服务器的文档根目录下, 并且填写表单, 输出将会如图14-9所示。但是, 你的结果很可能有所不同, 因为有很多变量起作用。

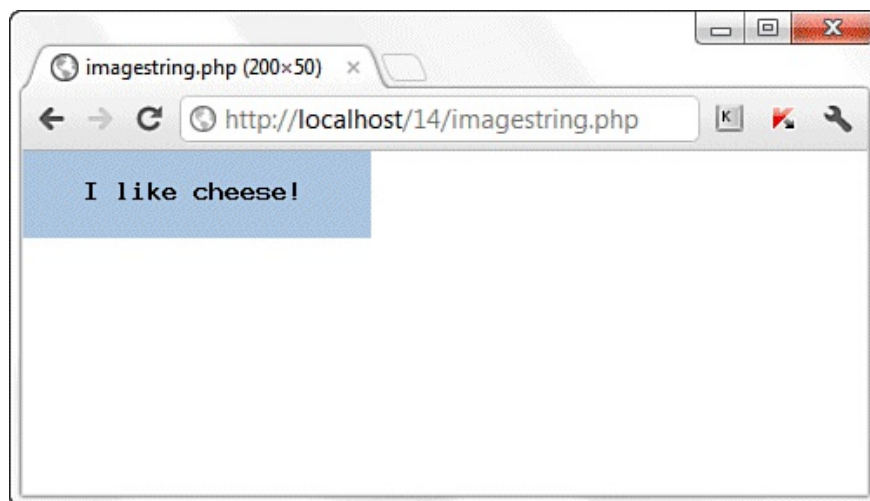


图14-9 图像创建表单的示例输出

用各种大小、颜色和文本字符串自己尝试一下。

## 14.7 使用脚本创建的图像

前面所有的脚本产生图像输出，但我们把它们当作单独的脚本调用。要在自己的HTML中使用脚本创建的图像的结果，这很简单，只需要在 `img` 标签的 `src` 属性中使用脚本（而不是图像）的名字，示例如下。

```

```

在程序清单14.8中，我们创建了一个简单的脚本来产生一幅图像。然后，用标签调用这个图像，并且在浏览器中显示它。

为了添加一些新的功能，这个基本脚本载入并使用一种定制字体来把文本字符串显示为一个图形，和前面小节中用户生成的文本很相似，只不过字体有趣一些。这么做的原因将很快会清楚。首先，创建生成图像的脚本。程序清单14.8给出了一个例子。

程序清单14.8 使用定制字体和文本创建一个图像



---

```
1: <?php
2: //create the canvas
3: $myImage = ImageCreate(150,25);
4:
5: //set up some colors for use on the canvas
6: $white = ImageColorAllocate($myImage, 255, 255, 255);
7: $black = ImageColorAllocate($myImage, 0, 0, 0);
8:
9: //load a font
10: $font = imageloadfont("hootie.gdf");
11:
12: // write the string
13: ImageString($myImage, $font, 0, 0, "CAPTCHA!", $black);
14:
15: //output the image to the browser
16: header ("Content-type: image/png");
17: ImagePng($myImage);
18:
19: //clean up after yourself
20: ImageDestroy($myImage);
21: ?>
```

---

第3行创建了一个宽150像素高25像素的画布。这个尺寸和我们通常在CAPTCHA中见到的内容大小相同。实际上，这就是我们所创建的内容的一部分。

**提示：**

CAPTCHA是一个图形化的挑战-响应测试，用来确定用户是否是人类而非机器。你可能会在下列情况下碰到过CAPTCHA：在完成一个表单，从而在站点上留下一条评论或参与一个讨论论坛的时候，或者在创建一个用户账户的时候。其思想是，只有人能够阅读图像中的字符文本。我们可以从<http://en.wikipedia.org/wiki/Captcha> 了解有关CAPTCHA的更多内容。

第5行到第7行为该脚本设置了颜色：白色作为背景颜色和黑色作为另一种颜色。第10行使用imageloadfont()函数载入字体。如果想要使用TrueType字体，也可以使用imagettftext()函数。在这个例子中，hootie.gdf字体是可以自由获取的字体，它取自于文件系统中的单个

文件，并且由调用它的函数载入。

**提示：**

我们可以通过在Google或者其他搜索引擎中搜索“free GDF font”或“free TrueType font”，以找到这个例子中使用的字体，以及其他的几种免费字体。

第13行使用第10行载入的字体作为ImageString()函数调用的一部分。在这个例子中，\$font是第二个参数，意味着包含在字体包中的字体和字体大小用于这个函数的实例中。开始的 X 和 Y 位置分别在第 3 个参数和第 4 个参数中直接编码为 0，并且将要绘制的字符串CAPTCHA!也是直接编码的。绘制这个字符串的颜色是\$black，这是最后一个参数的值。

第16~17行把图像数据流输出到Web浏览器：首先发送相应的header()函数，然后使用ImagePng()输出数据流；这个例子输出了一个PNG文件。第20行使用ImageDestroy()函数清除了脚本开始处的ImageCreate()函数所使用的内存。

如果在此时停下来，把这个脚本发送到 Web 服务器并直接在浏览器中载入它，将会以定制字体显示“CAPTCHA!”。但是，本节的目的是展示如何使用img标签来载入图像，以及展示需要一些HTML文件来完成什么。程序清单14.9包含了完成这项工作的一个img标签。

程序清单14.9 使用定制字体和文本创建一个图像

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>Using Images Created by Scripts</title>
5: </head>
```

---

```
6: </body>
7: <h1>Generated Image Below...</h1>
8: 
9: </body>
10: </html>
```

---

这个简短的HTML文件包含了有趣的一行，即第8行。在第8行，我们可以看到img标签的 src 属性的值是程序清单 14.8所创建的脚本的名字，即 imagecustomfont.php。把这个HTML文件取名为useimage.html，并把HTML和PHP脚本都发送给Web服务器。

在浏览器中打开useimage.html，你将会看到如图14-10所示的内容，即以定制字体显示的“CAPTCHA!”，这实际上是以一个图形显示的。

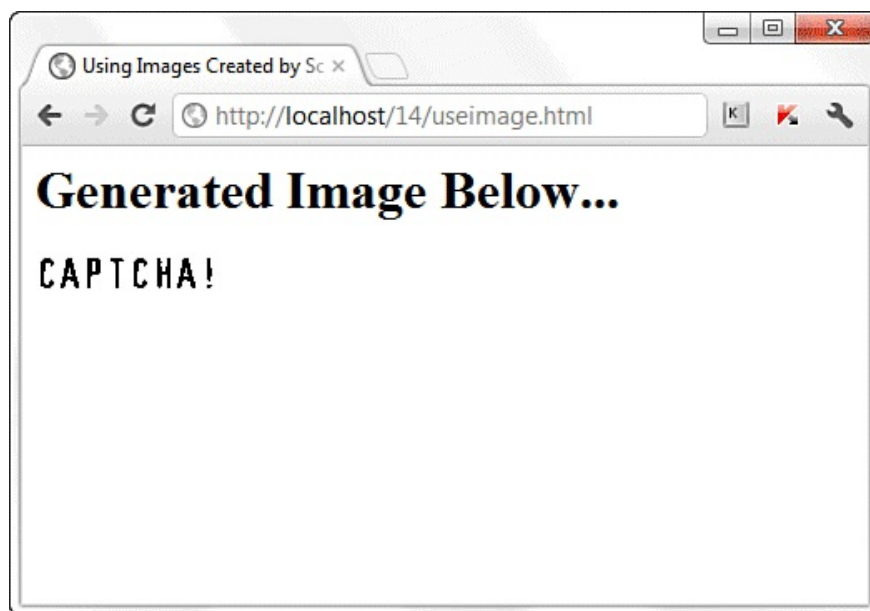


图14-10 使用脚本创建的图像

如果你真地使用这个脚本作为CAPTCHA系统的一部分，至少要对其做如下两方面的完善。随机化字符串，而不是对其直接编码；把随机产生的字符串保存在一个数据库表中，以便能够进行输入匹配过程。

## 14.8 小结

本章针对我们能够使用图像和 PHP 做什么给出了一个简短的介绍。我们强烈推荐位于<http://www.php.net/image> 的PHP手册的“Image Functions”部分，不仅因为这是和图像相关的函数的一个完整列表，而且这里还有很多有关它们在应用程序中的用法的例子和讨论。

在本章中，我们学习了安装和使用附加的库来操作图像，以 PNG 为例，但是，我们也给出了使用GIF和JPG的说明。我们学习了创建一个图像画布和分配颜色的基本步骤，以及绘制和填充图形的基本步骤。我们学习了绘制画布可以由一个已有的图像组成，还学习了合并层以使最终图像是一个合并了层的合成图像。另外，我们看到了在图像创建脚本中（在例子中是从一个 HTML 表单）使用用户输入是多么简单，以及如何在 HTML 代码中把脚本作为图像源使用。

## 14.9 Q&A

**Q:** 如何使用动态数据来创建一个分块的饼图？

**A:** 创建任何图形的时候，开始点和绘制长度都不需要静态指定，它们可以是变量，其值由数据库、用户输入或者脚本内的计算所确定。例如，下面这段代码创建一个红色的实心90度弧形。

```
ImageFilledArc($myImage,50,50,100,50,0,90,$red,IMG_ARC_PIE);
```

我们可以这样设置，使得位于饼图顶部的红色填充的弧形把May Sales占总数的百分比保存到一个名为\$may\_sales\_pct的变量中。然后，这行代码变为如下所示。

```
ImageFilledArc($myImage,50,50,100,50,0,$may_sales_pct,$red,IMG_ARC_PIE);
```

接着，数字将通过脚本中的计算或数据库查询而填入到函数。确保添加代码来验证所有弧度的总和为360度。

## 14.10 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 表示纯黑色和纯白色的RGB值是多少？
2. 如何创建一个新的、空白的、300像素宽、200像素高的画布？
3. 哪个函数用来绘制多边形？绘制一个填充的多变形呢？

## 解答

1. (0, 0, 0) 是纯黑色, (255, 255, 255) 是纯白色。
2. 使用如下命令, 创建一个新的、黑色的画布, 宽300像素, 高200像素。

```
$new_image = ImageCreate(300,200);
```

3. ImagePolygon()和ImageFilledPolygon()。



## 思考题

1. 使用绘画技能创建一个条状图替换饼图，条状图带有垂直或水平的条块，每个 30 像素宽，并且用同样的颜色填充，但它们具有不同的长度。确保每个条块之间有 10 个像素的空间。
2. 扩展思考题 1 的功能，在其前端添加一个表单，允许用户输入自己选择的数值。根据这些数值来生成条状图。

## 第4部分 PHP与MySQL整合

第15章 理解数据库设计过程

第16章 SQL基本命令

第17章 使用MySQL中的事务和存储过程

第18章 使用PHP和MySQL交互

## 第15章 理解数据库设计过程

本章所涉及的主题包括：

- 良好的数据库设计的一些优点。
- 表关系的三种类型。
- 如何规范化数据库。
- 如何实现一个良好的数据库设计过程。

在本章中，我们将学习设计一个关系数据库背后的过程。在这个关注理论的一章中，我们直接开始学习基本的MySQL命令，为把MySQL整合到自己的应用程序中而做准备。

## 15.1 良好的数据库设计的重要性

良好的数据库设计对于一个高性能的应用程序非常重要，就像一个空气动力装置对于一辆赛车的重要性一样。如果一辆汽车没有平滑的曲线，将会产生阻力从而变慢。关系没有经过优化，数据库无法尽可能高效地运行。应该把数据库的关系和性能（包括易维护性、最小化、可重复性以及避免不一致性）看作是规范化的一部分。

### 提示：

规范化指的是为了尽量避免重复性和不一致性而组织数据结构的过程。

除了性能以外的问题，就是维护的问题了，数据库应该易于维护。这包括只存储数量有限的（如果有的话）重复性数据。如果有很多重复性数据，这些数据的一个实例发生一次改变（例如，一个名字的改变），这个改变必须对所有的其他的数据都进行。为了避免重复，并且增强维护数据的能力，我们可以创建可能的值的一个表并使用一个键来引用该值。在这种方式中，如果值改变了名字，这个改变只在主表中发生一次，所有的其他表的引用都保持不变。

例如，假设你负责维护一个学生数据库以及他们所注册的课程。如果有35个学生在同一个课堂中，让我们将这门课叫做Advanced Math（高等数学），课程的名字将会在表中出现35次。现在，如果老师决定把这门课的名字改为Mathematics IV，我们必须修改35条记录以反映出新的课程名。如果数据库设计为课程名出现在一个表中，只有课程

ID号码和学生记录一起存储，那么要更改课程名称，我们就只需要改变一条记录而不是 35 条记录。

一个规划和设计良好的数据库的优点是众多的，它也证实了这样一个道理，前期做的工作越多，后面所要做的就越少。在使用数据库的应用程序公开发布之后，还要对数据库进行重新设计，这是最糟糕的，然而，这确实会发生，并且代价高昂。

因此，在开始编写一个应用程序的代码之前，请花大量的时间来设计你的数据库。在本章其余的部分中，我们将学习很多有关关系和规范化的内容，这是设计难题中最重要的两部分。

## 15.2 表关系的类型

表关系具有以下几种形式。

- 一对一关系。
- 一对多关系。
- 多对多关系。

例如，假设有一个名为**employees**的表，其中包含了每个人的社会安全号码、名字和他所在的部门。假设我们还有一个名为**departments**的表，其中包含了所有部门的列表，每个部门有一个部门ID和一个名字。在**employees**表中，部门ID字段和**departments**表中的ID字段相对应。我们可以在图15-1中见到这种类型的关系。字段名旁边的PK表示该表的主键。

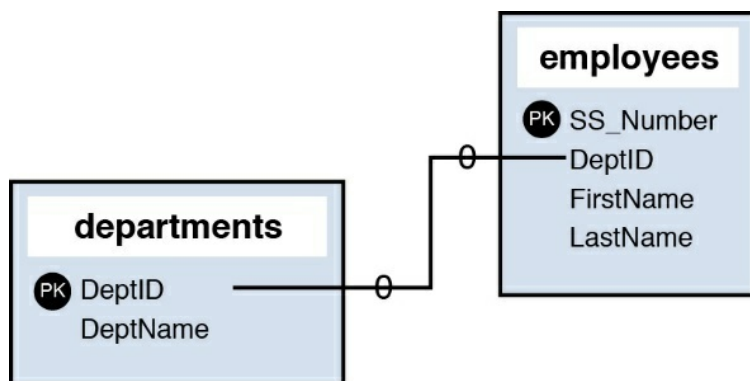


图15-1 employees表和departments表通过DeptID键联系起来

在下面的小节中，我们将进一步详细介绍每种关系。

### 15.2.1 一对一关系

在一对一的关系中，一个键只能在一个关系表中出现一次。  
employees表和departments表没有一对一的关系，因为显然很多职员都属于同一个部门。但是，如果公司中的每个职员分配了一台计算机，这就存在一对一的关系了。图15-2给出了职员和计算机之间的一对一的关系。

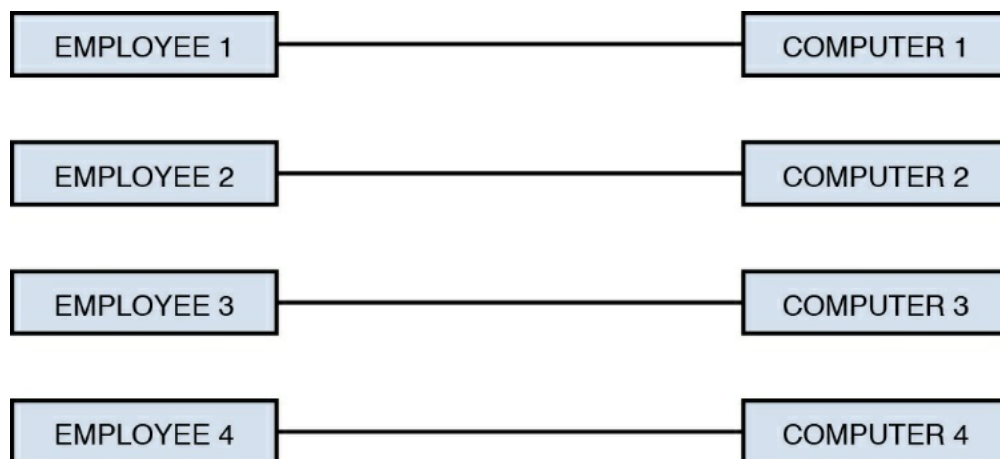


图15-2 每个职员分配一台计算机

数据库中的employees表和computers表看上去如图15-3所示，表示了它们一对一的关系。

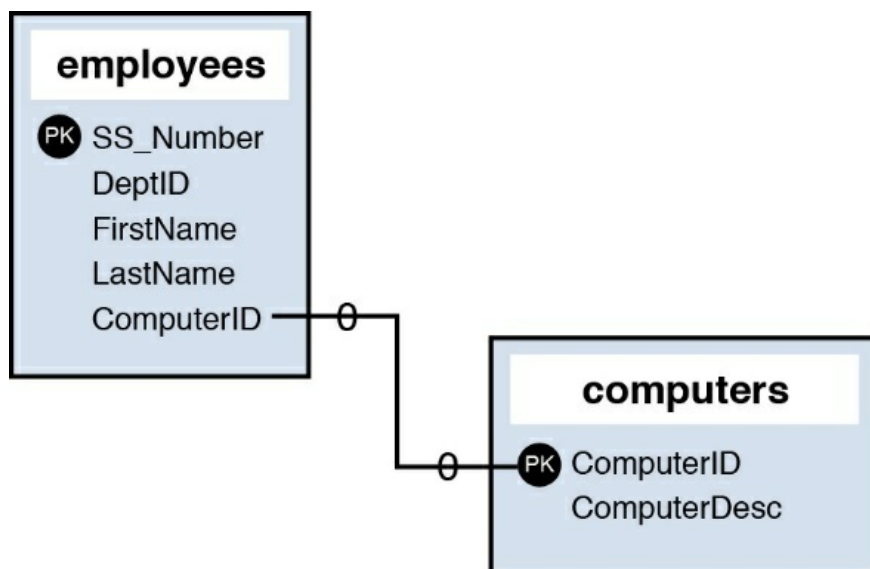


图15-3 数据模型中的一对一关系

### 15.2.2 一对多关系

在一个一对多关系中，一个表中的键在一个相关的表中出现多次。如图15-1中的例子所示，它表示的是职员和部门之间的关系，即一个一对多的关系。现实中的一个例子就是部门的组织图，如图15-4所示。

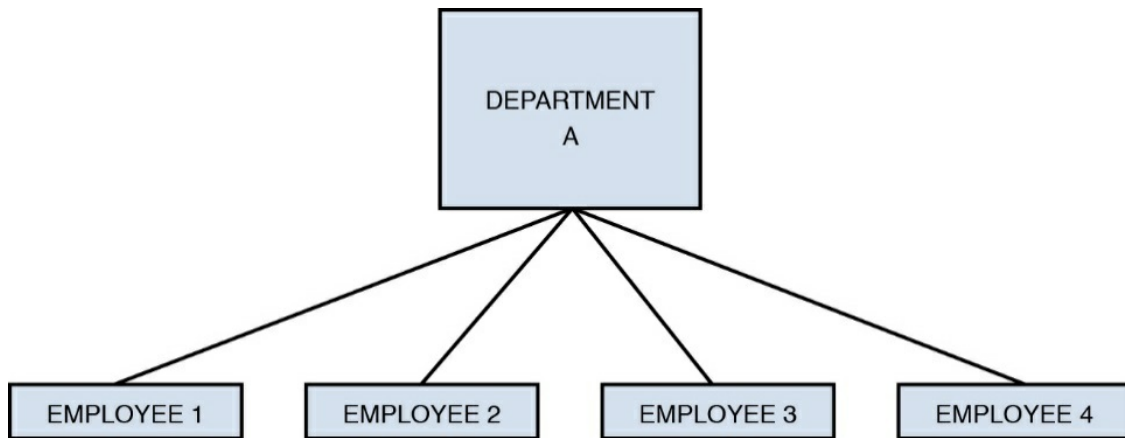


图15-4 一个部门包含了多个职员

一对多的关系是最为常见的一种关系。另一个实际的例子就是在一个地址数据库中使用一个州的缩写，每个州都有一个唯一的标识符（CA表示California、PA表示Pennsylvania等），并且美国的每个地址都有一个相关的州。

如果有8个朋友在California并且有5个朋友在Pennsylvania，在我们的表中将只使用两个不同的缩写。一个缩写（CA）表示一个一对八的关系，另一个缩写（PA）表示一个一对五的关系。

### 15.2.3 多对多关系

多对多关系通常会在规范化的数据库的实际例子中引发问题，以至



于通常直接把多对多关系分解为一系列的一对多关系。在多对多关系中，一个表的键值可以在一个相关的表中出现多次。因此，它听上去就像是一个一对多关系；但是反之亦然，即第二个表的主键也可以在第一个表中出现多次。

按照这样一种方式来思考一种关系，可以以学生和课程为例：一个学生有一个ID和一个名字；一门课程有一个ID和一个名字。正如图15-5所示，一个学生可能一次选多门课程；而一门课程总是有一个以上的学生。

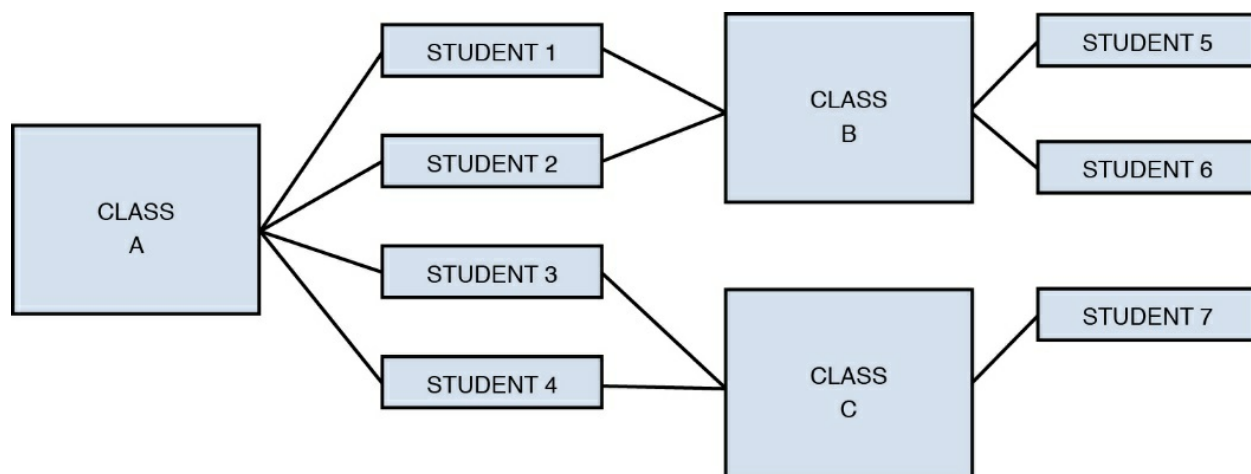


图15-5 学生选择课程，课程包含学生

正如你所看到的，这种关系不能以一种相关表的简单方法表示。这个表也可能看上去如图15-6所示，似乎是不相关的。



图15-6 学生表和班级表不相关

为了让多对多关系更理论化，我们可能创建一个中间表，这个表位于两个表之间，并且实际上把它们映射到一起。我们可以构建和图15-7中所示的表类似的一个表。

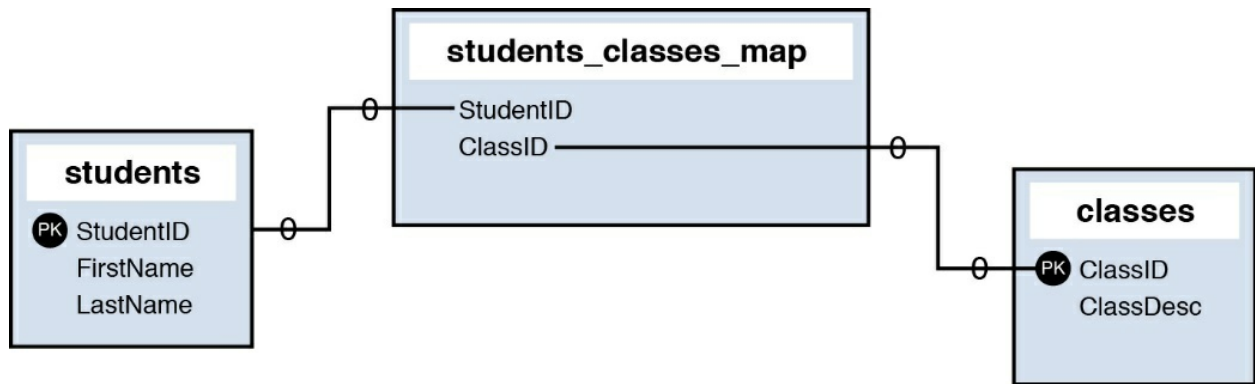


图15-7 students\_classes\_map表充当一个中间表

如果取出图15-5中的信息并且将其放入到这个中间表中，我们会得到如图15-8所示的结果。

STUDENTID	CLASSID
STUDENT 1	CLASS A
STUDENT 2	CLASS A
STUDENT 3	CLASS A
STUDENT 4	CLASS A
STUDENT 5	CLASS B
STUDENT 6	CLASS B
STUDENT 7	CLASS C
STUDENT 1	CLASS B
STUDENT 2	CLASS B
STUDENT 3	CLASS C
STUDENT 4	CLASS C

图15-8 填充了数据的students\_classes\_map表

可以看到，多个学生和多门课程在students\_classes\_map 表中愉快地共存。

在介绍了关系的类型之后，我们再来学习规范化就是小菜一碟了。

## 15.3 理解规范化

规范化只是一组规则，当我们作为一名数据库管理员的时候，这套规则会让我们的生活很容易。按照这样一种方式来组织数据库，使得其相关的表能够适应和灵活应对未来的增长，这是一门技艺。

规范化中用到的这一套规则叫做范式。如果数据库设计遵从了第一组规则，它就考虑使用第一范式。如果遵从了前3组规则，就说明我们的数据库使用了第三范式。

在整个本章中，我们将学习第一范式、第二范式和第三范式中的每条规则。我们希望，在你创建自己的应用程序的时候能够遵从它们。下面还将使用学生和课程数据库的一组示例表，并且用第三范式规范它们。

### 15.3.1 平表带来的问题

在开始学习第一范式之前，我们必须从需要修复的地方开始。在一个数据库的例子中，这就是一个平表（flat table）。平表就像是一个表单，它有很多很多的列。多个表之间没有关系，我们需要的所有数据都可能在这个平表之中。这是一种没有效率的情景，而且会比规范化的数据库消耗掉更多的硬盘物理空间。

在学生和课程数据库中，假设在平表中有如下的字段。

- **StudentName** ——学生的名字。
- **CourseID1** ——学生选择的第一门课程的ID。

- **CourseDescription1** ——学生选择的第一门课程的说明。
- **CourseInstructor1** ——学生选择的第一门课程的老师。
- **CourseID2** ——学生选择的第二门课程的ID。
- **CourseDescription2** ——学生选择的第二门课程的说明。
- **CourseInstructor2** ——学生选择的第二门课程的老师。
- 针对学生在他们的学业中涉及的所有课程，CourseID、CourseDescription和CourseInstructor列可以多次重复出现。

根据我们目前已经学习的内容，我们应该能够识别第一个问题：CourseID、CourseDescription和CourseInstructor列是重复的组。

去除冗余是规范化的第一步，因此，接下来我们要把平表纳入到第一范式中。如果我们的表保留了平的格式，我们可能有很多未占用的空间并且很多空间都没必要使用，这不是一个有效率的表设计。

### 15.3.2 第一范式

第一范式的规则如下。

- 去除重复的信息。
- 为相关的数据单独创建一个表。

如果考虑学生和课程数据库使用带有很多重复的字段组的平表设计，我们可以区分两个明显的主题，即学生和课程。把学生和课程数据库纳入到第一范式，意味着我们要创建两个表，一个用于学生，一个用于课程，如图15-9所示。

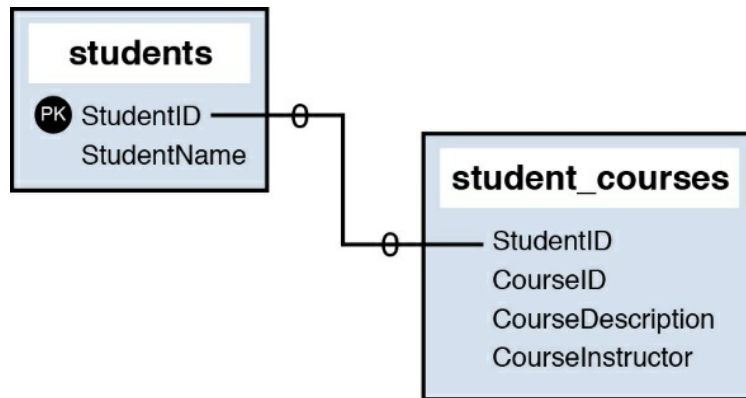


图15-9 把平表分解为两个表

两个表现在表示了一个学生到多门课程的一对多的关系。学生可以选取尽可能多的课程，而不会受到平表中已有的 CourseID/CourseDescription/CourseInstructor 组的数目的限制。

下一步把这些表纳入第二范式。

### 15.3.3 第二范式

第二范式的规则如下：

- 没有依赖于主键的真子集的非主键属性。

用通俗易懂的语言来说，这意味着，如果表中的字段不是完全和一个主键相关，我们有更多的工作要做。在学生和课程的例子中，我们需要把课程分解到他们自己的表中并且修改 students\_courses 表。

CourseID、CourseDescription 和 CourseInstructor 可能成为一个叫做 courses 的表，它拥有一个 CourseID 主键。随后，students\_courses 表应该只包含两个字段 StudentID 和 CourseID。我们可以在图 15-10 中看到这个新的设计。

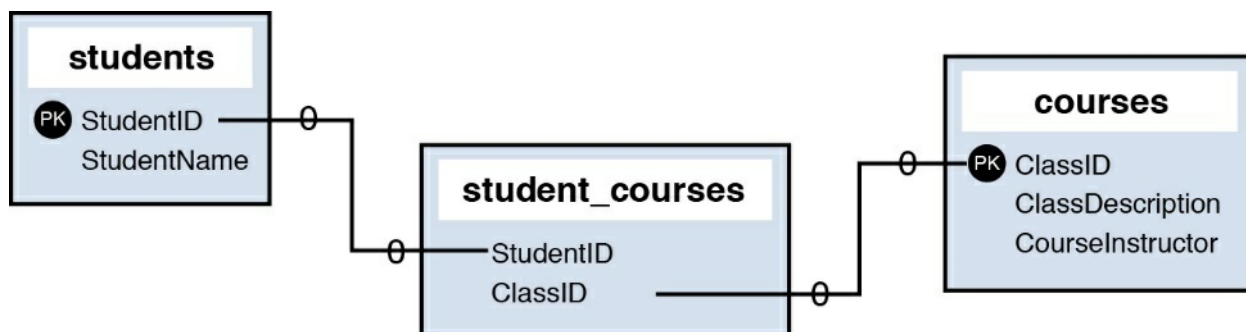


图15-10 使表符合第二范式

这个结构我们应该熟悉了，就像在一个多对多的关系中使用一个中间映射表那样。第三范式是我们将要学习的最后一个范式，你将发现，它理解起来和前两个范式一样简单。

### 15.3.4 第三范式

第三范式的规则如下所示。

- 没有依赖于非主键属性的属性。

这条规则只是意味着我们需要查看表，看看表是否有更多的字段可以进一步分解，以便可以不再依赖于一个键。考虑删除重复的数据，我们将找到答案，这就是教师。不可避免的，一个教师将要教授多门课程。然而，CourseInstructor不是任何类型的一个主键。因此，如果我们把这个信息分解并且纯粹为了效率和可维护性来创建一个单独的表（如图15-11所示），这就是第三范式。

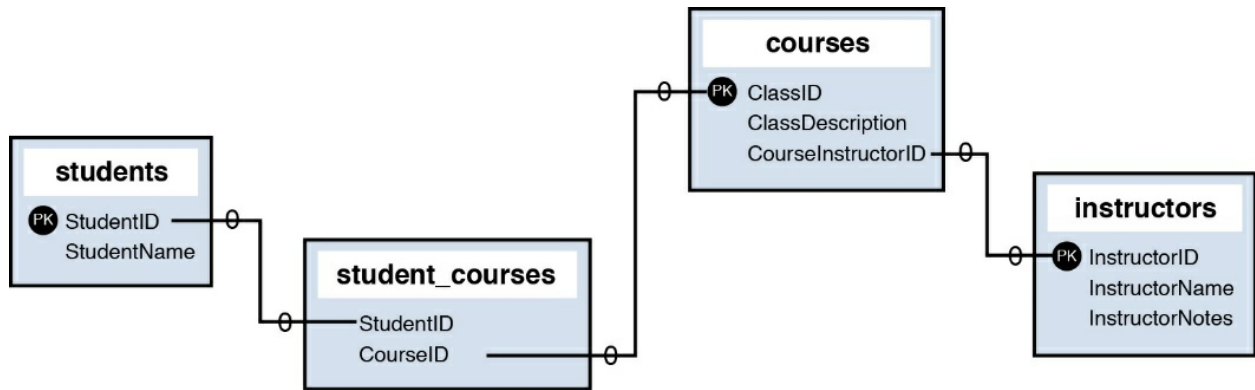


图15-11 把表纳入第三范式

第三范式通常删除了常见的冗余并且考虑到了灵活性和增长性。下面将针对涉及到数据库设计的思考过程以及数据库设计在何处加入到应用程序的全面设计过程中给出一些指南。



## 15.4 遵从设计过程

应用程序设计中的最大问题就是缺乏前期思考。这也适用于数据库驱动的应用程序，设计过程必须包含对数据的全面考虑，这应当包括数据应该如何彼此相关。最重要的是，数据是否是可扩展的。

设计过程中的一般步骤如下。

- 定义目标。
- 设计数据结构（表、字段）。
- 分清关系。
- 定义和实现业务规则。
- 创建应用程序。

创建应用程序是最后一步，而不是第一步。很多开发者从应用程序开始构建它，然后回过头来试图构建一组数据库表来填入应用程序的数据。这种方法完全是南辕北辙，效率低下，并且花费很多的时间和金钱。

在我们开始任何应用程序设计过程之前，坐下来进行详细的讨论。如果我们不能描述应用程序，包括目标、用户和目标市场，那么，我们还没有准备好构建它，更不要说对数据库建模了。

在可以向其他人描述应用程序的功能和差别，并且使这对他们有意义之后，我们可以开始考虑想要创建的表了。首先从一个大的平表开始，因为在创建好平表之后，我们刚刚学习的规范化技巧才有了用武之

地。我们将能够找到冗余，并且把关系可视化。随着你越来越有经验，你将能够把这个过程的步骤最小化，但是，仔细而明确地经过这些步骤也不为过。

下一步就是进行规范化。从平表到第一范式，然后，可能的话继续升级到第三范式。使用纸、铅笔、记事贴等任何能够有助于可视化表和关系的工具。在你准备自己创建表之前，在记事贴上进行数据建模，这没什么丢人的。另外，使用记事贴比购买软件来进行建模要便宜很多，建模软件的价格从数百美元到数千美元不等。

在我们有了一个初步的数据模型之后，从应用程序的角度看看它，或者从我们所构建的应用程序的用户的角度来看看它。我们也正是从这个角度来定义业务规则并且看看数据模型是否违反了规则。例如，一个在线注册应用程序的一条业务规则是“每个用户必须有一个Email地址，并且它必须不属于任何其他已有的用户”。如果EmailAddress在你的数据模型里不是一个唯一的字段，你的模型将会违反这一业务规则。

在业务规则已经应用到数据模型后，只有这时候才可以开始应用程序编程。只要确保数据模型是健壮的，我们就可以安心了，不需要把自己也加入到程序编写中去。后续的事情没什么特别的。

## 15.5 小结

遵照正确的数据库设计是确保应用程序能够高效、灵活并且易于管理和维护的唯一方法。数据库设计的一个重要方面就是使用表之间的关系，而不是把所有的数据都一股脑放到一个长长的平表文件中。关系的类型包括一对一关系、一对多关系和多对多关系。

使用关系来恰当地组织数据，这叫做规范化。有很多层级的规范化，但是主要的层级是第一范式、第二范式和第三范式。每个层级都有必须遵守的一条或两条规则。遵守所有的这些规则将确保我们的数据库设计良好而且灵活。

要让一个思想从刚刚起步到产生成果，我们需要遵从一个设计过程。这个过程通常是“三思而后行”。讨论规则、需求以及目标，然后创建规范化表的最后版本。

## 15.6 Q&A

**Q:** 只有3种范式？

**A:** 不是的。有多种范式。其他的范式是 Boyce-Codd 范式、第四范式和第五范式（也称为Join-Projection范式）。在实际的应用程序开发中，通常并不使用这些范式，因为遵守这些范式需要付出的人力和数据库效率代价太大（但如果你实现它们的话，有可能很不错）。要了解更多信息，请访问

[http://en.wikipedia.org/wiki/Database\\_normalization#Normal\\_forms](http://en.wikipedia.org/wiki/Database_normalization#Normal_forms) 。

## 15.7 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 说出3种数据关系类型。
2. 由于多对多的关系在高效率的数据库设计中很难表示，我们该怎么做呢？
3. 说出创建数据关系可视化的几种方式。

## 解答

1. 一对一关系、一对多关系和多对多关系。
2. 使用中间映射表来创建一系列一对多的关系。
3. 你可以使用各种工具，从记事贴和字符串（注意，其中是表格以及表示表格之间的关系的字符串），到用于绘图的软件，到说明SQL语句和提供可视化的软件程序。

## 思考题

向使用表单和平表的一个入解释全部3种范式。



## 第16章 SQL基本命令

在本章中，我们将学习：

- 基本的MySQL数据类型。
- 如何使用**CREATE TABLE**命令来创建表。
- 如何使用**INSERT**命令来输入记录。
- 如何使用**SELECT**命令来获取记录。
- 如何在**SELECT**表达式中使用基本函数、**WHERE**子句以及**GROUP BY**子句。
- 如何使用**JOIN**或子查询从多个表中查询。
- 如何使用**UPDATE**和**REPLACE**命令来修改已有记录。
- 如何使用**DELETE**命令来删除记录。
- 如何使用MySQL内建的字符串函数。
- 如何使用MySQL内建的日期和时间函数。

上一章已经介绍了数据库设计过程的基础知识，本章介绍核心SQL语法的初步知识，我们将会使用这些语法来创建和操作MySQL数据库表。这是动手实践的一章，并且假设你能够直接向MySQL发布命令，要么通过MySQL命令行界面，或者通过phpMyAdmin这样的管理界面。在本书第1章的Quick Start过程的XAMPP安装中包含了phpMyAdmin。

请注意，这可能不是本书中最令人激动的一章，但它将展示很多元素的操作示例，你将在以后的工作中用到这些元素。

## 16.1 MySQL数据类型

在表中正确地定义字段对于全面优化数据库很重要。我们应该对字段只使用真正需要使用的类型和大小，如果我们只需要使用两个字符，就不要把一个字段定义为10个字符的宽度，数据库必须考虑那8个额外的字符，即便不会使用它们。这些字段的类型也叫做数据类型，因为我们将要在这些字段中存入“该类型的数据”。

MySQL使用多种不同的数据类型，这些数据类型可以划分为3类：数字类型、日期和时间类型、以及字符串类型。请密切注意它们，因为定义数据类型比表创建过程中的任何其他部分都更为重要。

### 16.1.1 数字数据类型

MySQL使用所有标准的ANSI SQL数字数据类型，因此，如果我们从其他的数据库系统转到MySQL，这些定义对你来说会很熟悉。如下的列表给出了常用的数字数据类型以及其说明。

#### 提示：

数字数据类型的列表中将用到有符号或者无符号这样的术语。如果你还记得基本的代数知识，就会知道一个有符号整数可以是一个正整数或者负整数，而一个无符号的整数总是一个非负的整数。

- **INT** ——一个常规大小的整数，可以是有符号的或者无符号的。如果是有符号的，允许的范围从-2147483648到2147483647。如果是无符号的，允许的范围从0到4294967295。我们可以指定最大11位

的宽度。

- **TINYINT** ——一个小的整数，可以是有符号的或者无符号的。如果是有符号的，允许的范围从-128到127。如果是无符号的，允许的范围从0到255。我们可以指定最大 4位的宽度。
- **SMALLINT** ——一个小的整数，可以是有符号的或者无符号的。如果是有符号的，允许的范围从-32768到32767。如果是无符号的，允许的范围从0到65535。我们可以指定最大5位的宽度。
- **MEDIUMINT** ——一个中等大小的整数，可以是有符号的或者无符号的。如果是有符号的，允许的范围从-8388608到8388607。如果是无符号的，允许的范围从0到16777215。我们可以指定最大9位的宽度。
- **BIGINT** ——一个较大的整数，可以是有符号的或者无符号的。如果是有符号的，允许的范围从-9223372036854775808到9223372036854775807。如果是无符号的，允许的范围从0到18446744073709551615。我们可以指定最大11位的宽度。
- **FLOAT(M,D)** ——一个浮点数，不能是无符号的。我们可以定义显示长度（M）和小数位长度（D）。但这不是必须的，默认值为10,2，其中2是小数位数，而10是总位数（包括小数位）。一个FLOAT的小数位精度可以达到24位。
- **DOUBLE(M,D)** ——一个双精度浮点数，不能是无符号的。我们可以定义显示长度（M）和小数位长度（D）。但这不是必须的，默认值为16,4，其中4是小数位数。一个DOUBLE的小数位精度可以达到53位。REAL是DOUBLE的同义词。
- **DECIMAL(M,D)** ——一个未打包的双精度浮点数，不能是无符号的。在未打包的小数中，每个小数对应一个字节。定义显示长度

(M) 和小数位长度 (D) 是必须的。NUMERIC是DECIMAL的同义词。

在所有的MySQL数字数据类型中，最经常使用INT。如果我们定义自己的字段比实际所需的小，可能会遇到问题。例如，如果我们把一个ID字段定义为无符号的TINYINT，如果ID是一个主键（并且是必须的字段），我们将不能成功地插入第256条记录。

### 16.1.2 日期和时间类型

MySQL有几种数据类型可以用来存储日期和时间，这些数据类型在输入方面很灵活。换句话说，我们可以输入那些并不是真正的日子的日期，例如2月30号，而2月只有28或29号，没有30号。另外，我们可以存储带有遗失的信息的日期。例如，如果我们知道某人出生于1980年11月的某天，可以使用1980-11-00，而00表示出生的日期，如果我们知道日期的话。

MySQL的日期和时间类型的灵活性也意味着，日期检查的职责落到了应用程序开发者的肩上。MySQL只检查两个元素的有效性：月份是否在0到12之间，以及日期在0到31之间。MySQL不会自动验证2月的30号是否是有效的日期。因此，应用程序内所要进行的任何日期验证，都应该在PHP代码中进行，而且在试图用假的日期向数据库表添加一条记录之前就进行验证。

MySQL的日期和时间数据类型如下。

- **DATE** ——YYYY-MM-DD格式的一个日期，在1000-01-01到9999-12-31之间。例如，1973年12月30日，将存储为1973-12-30。

- **DATETIME** ——YYYY-MM-DD HH:MM:SS格式的一个日期和时间组合，在1000-01-01 00:00:00到9999-12-31 23:59:59之间。例如，1973年12月30日下午3:30将存储为1973-12-30 15:30:00。
- **TIMESTAMP** ——1970年1月1日午夜和2037年某个时间之间的一个时间戳。我们可以为TIMESTAMP定义多个长度，这直接和其中存储的内容相关。TIMESTAMP缺省的长度是14，其中存储了YYYYMMDDHHMMSS。这看上去和前面的DATETIME格式相似，只是在数字之间没有连字符号。1973年12月30日下午3:30，将存储为19731230153000。TIMESTAMP的其他定义是12 (YYMMDDHHMMSS)、8 (YYYYMMDD)和6 (YYMMDD)。
- **TIME** ——以HH:MM:SS格式存储时间。
- **YEAR(M)** ——以2位或4位的格式存储年份。如果长度指定为2（例如YEAR(2)），YEAR可以是1970到2069（70到69）。如果长度指定为4，YEAR可以是1901到2155。默认长度是4。

DATETIME或DATE比其他的日期和时间相关的数据类型更为常用。

### 16.1.3 字符串类型

尽管数字和日期类型很有趣，但我们所存储的大多数数据将是字符串格式的。这里列出了MySQL中常用的字符串数据类型。

- **CHAR(M)** ——一个定长的字符串，长度在1到255个字符之间，例如CHAR(5)。存储的时候，右边使用空白填充到指定的长度，定义时长度不是必须的，但默认为1。
- **VARCHAR(M)** ——一个变长的字符串，长度在1到255个字符之

间，例如VARCHAR(25)。在创建一个VARCHAR字段的时候，必须定义一个长度。

- **BLOB 或TEXT** ——最大长度为65535个字符的一个字段。BLOB表示“Binary Large Objects”（二进制大对象），并且用来存储大量容量的二进制数据，例如图像或者其他类型的文件。定义为TEXT的字段也存储大量的数据。二者之间的不同在于，对于存储的数据的排序和比较，在BLOB上是区分大小写的，而在TEXT字段上是不区分大小写的。我们不对BLOB或TEXT指定长度。
- **TINYBLOB 或TINYTEXT** ——最大长度为255个字符的一个BLOB或TEXT。我们不对TINYBLOB或TINYTEXT指定一个长度。
- **MEDIUMBLOB 或MEDIUMTEXT** ——最大长度为16777215个字符的一个BLOB或TEXT。我们不对MEDIUMBLOB或MEDIUMTEXT指定一个长度。
- **LOB 或LONGTEXT** ——最大长度为4294967295个字符的一个BLOB或TEXT。我们不对LOB或LONGTEXT指定一个长度。
- **ENUM** ——一个枚举类型，即指定项目的一个列表。当定义一个ENUM的时候，我们创建了一个项目的列表，值必须从这个列表选定或者为NULL。例如，如果希望自己的字段包含“A”或“B”或“C”，我们可以定义ENUM为ENUM('A', 'B', 'C')，则只有这些值或者NULL可以填入到这个字段。ENUM可以有65535个不同的值。ENUM使用一个索引来存储项目。

提示：

SET类型和ENUM类型相似，因为它也定义了一个列表。然而，SET类型存储为一个完整

的值，而不是像ENUM那样，存储为一个值的一个索引。

VARCHAR和TEXT字段比其他的字段类型更为常用，而ENUM也很有用。

## 16.2 表的创建语法

表创建命令需要如下几个要素。

- 表的名字。
- 字段的名字。
- 每个字段的定义。

创建表的一般语法如下。

```
CREATE TABLE table_name (column_name column_type);
```

表名取决于你，但是应该是能够反应表的用途的名字。例如，如果你有一个表用来存储一个杂货店的存货，不应该把这表命名为s而应该将其命名为类似grocery\_inventory的名字。类似的，我们所选择的字段名也应该尽可能地精炼，并且和它们所起的作用以及它们所存储的数据相关。例如，我们可能把保存商品名字的一个字段命名为item\_name，而不是n。

下面的例子创建了一个通用的grocery\_inventory表，它保存了用于商品ID、商品名称、商品介绍、定价和数量的字段。这些字段的数据类型各不相同，ID和数量字段保存整数，商品名称字段最多保存50个字符，商品介绍字段最多保存65535个文本字符，而定价字段保存一个浮点数，示例如下。



```
CREATE TABLE grocery_inventory (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    item_name VARCHAR (50) NOT NULL,  
    item_desc TEXT,  
    item_price FLOAT NOT NULL,  
    curr_qty INT NOT NULL  
);
```

**提示:**

id字段定义为一个主键。我们将在后面的章节中，即在把具体的表作为示例应用程序的一部分创建的环境中，学习有关键的内容。通过使用auto\_increment作为字段属性，我们告诉MySQL继续前进并且把下一个可用的编号作为ID字段的值。

MySQL服务器使用Query OK响应一条成功执行的命令，而不管这条命令的类型是什么。否则，将会显示一条错误消息，告诉你查询出错。根据你所使用的界面的不同，你可能会也可能不会看到这条特定的响应。然而，不管使用什么界面，它应该都会给出和查询状态相关的提示。

## 16.3 使用INSERT命令

在创建了一些表后，可以使用SQL命令INSERT来向这些表添加新的记录。INSERT的基本语法如下。

```
INSERT INTO table_name (column list) VALUES (column values);
```

在括号中的值列表中，我们必须使用引号括起字符串。SQL标准是单引号，但MySQL允许使用单引号或者双引号。如果引号在字符串本身之中，别忘了对所用的引号的类型进行转义。

提示：

整数不需要使用引号括起来。

下面是一个需要转义的字符串的例子。

```
O'Connor said "Boo"
```

如果我们把字符串放入到双引号中，INSERT语句将会如下所示。

```
INSERT INTO table_name (column_name) VALUES ("O'Connor said \"Boo\"");
```

如果我们把字符串放入到一个单引号中，INSERT语句将会如下所示。

```
INSERT INTO table_name (column_name) VALUES ('O\'Connor said "Boo"');
```

### 进一步学习INSERT语句

除了表的名字，INSERT语句中还有两个重要的部分，即列列表和

值列表。只有值列表是必须的，但是，如果省略了列列表，必须严格按照对应的顺序在值列表中为这些列指定值。

以grocery\_inventory表为例，我们有5个字段：id、item\_name、item\_desc、item\_price和curr\_qty。要插入一条完整的记录，可以使用如下这些语句中的任何一条。

- 带有所有列名的一条语句，示例如下。

```
INSERT INTO grocery_inventory  
(id, item_name, item_desc, item_price, curr_qty)  
VALUES ('1', 'Apples', 'Beautiful, ripe apples.', '0.25', 1000);
```

- 使用所有列，但不显式指定这些列的一条语句，示例如下。

```
INSERT INTO grocery_inventory VALUES ('2', 'Bunches of Grapes',  
'Seedless grapes.', '2.99', 500);
```

尝试一下这两条语句，看看会发生什么。两条语句都应该产生“Query OK”结果。

现在，介绍使用INSERT的一些更为有趣的方法。由于ID是一个自动增加的整数，我们不必将其放入到值列表中。然而，如果有一个值我们有意不想列出，例如id，那必须把后续使用到的其他列都列出来。例如，如下的语句没有给出列列表，并且也没有给id一个值。

```
INSERT INTO grocery_inventory VALUES  
('Bottled Water (6-pack)', '500ml spring water.', 2.29, 250);
```

上述语句或产生如下的一个错误。

```
ERROR 1136: Column count doesn't match value count at row 1
```

由于没有列出任何列，MySQL期待它们都出现在值列表中，从而

导致前面一条语句产生错误。如果目标是让MySQL通过自动增加id字段来为你完成工作，我们可以使用下面这些语句中的一条。

- 带有除了id以外的所有列名的一条语句。

```
INSERT INTO grocery_inventory (item_name, item_desc, item_price, curr_qty)  
VALUES ('Bottled Water (6-pack)', '500ml spring water.', '2.29', 250);
```

- 使用所有的列，但是，不显式给出它们的名字，并且对id使用一个NULL值以便MySQL可以自动填入一个值。

```
INSERT INTO grocery_inventory VALUES ('NULL', 'Bottled Water (12-pack)',  
'500ml spring water.', 4.49, 500);
```

这两条语句都尝试一下，这样grocery\_inventory表一共就有了4条记录了。选择哪条语句对于MySQL来说没有什么区别，这是你个人的偏好而已，但是应在你的应用程序开发中保持一致。一致的结构将会使你随后的调试工作更容易，因为你将知道预期得到什么结果。

## 16.4 使用SELECT命令

SELECT是用来从表中获取数据的命令。这条命令的语法可以非常简单也可能非常复杂，取决于你所要选择的字段，你是否从多个表中选取，以及你计划施加什么条件。随着对数据库程序设计越来越熟悉，你将会学习扩展SELECT语句，最终使得你的数据库做尽可能多的工作而不会让编程语言负担过重。

最基本的SELECT语法如下所示。

```
SELECT expressions_and_columns FROM table_name  
[WHERE some_condition_is_true]  
[ORDER BY some_column [ASC | DESC]]  
[LIMIT offset, rows]
```

看看其中第一行，内容如下。

```
SELECT expressions_and_columns FROM table_name
```

一个方便的表达式就是\*符号，它表示一切内容。因此，要查询grocery\_inventory表中的一切内容（所有行、所有列），SQL语句如下。

```
SELECT * FROM grocery_inventory;
```

根据在grocery\_inventory中找到的数据的多少，我们的结果可能各不相同，但是，结果会如下所示。

```

+-----+-----+-----+-----+
| id | item_name          | item_desc          | item_price | curr_qty |
+-----+-----+-----+-----+
| 1 | Apples             | Beautiful, ripe apples. | 0.25      | 1000     |
| 2 | Bunches of Grapes  | Seedless grapes.      | 2.99      | 500      |
| 3 | Bottled Water (6-pack) | 500ml spring water.  | 2.29      | 250      |
| 4 | Bottled Water (12-pack) | 500ml spring water.  | 4.49      | 500      |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)

```

这是MySQL命令行界面的输出，你可以看到，MySQL创建了一个可爱的、格式化的表，使用结果集的第一行作为表的列名。如果你使用不同的MySQL界面，结果看上去会有所不同（注意观察期望的数据，而不是界面的不同）。

如果我们只想选择特定的列，使用列名来替代\*，多个列名之间用逗号隔开。如下的语句只从grocery\_inventory表选择id、item\_name和curr\_qty字段。

```
SELECT id, item_name, curr_qty FROM grocery_inventory;
```

结果显示如下所示。

```

+-----+-----+-----+
| id | item_name          | curr_qty |
+-----+-----+-----+
| 1 | Apples             | 1000     |
| 2 | Bunches of Grapes  | 500      |
| 3 | Bottled Water (6-pack) | 250      |
| 4 | Bottled Water (12-pack) | 500      |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

### 16.4.1 排序SELECT结果

SELECT查询的结果默认按照它们插入到表中的顺序来排序，并且不会依赖于一个有意义的排序系统。如果我们想要按照一种特定方式对结果排序，例如，按照日期、ID、名字等等来排序，需要使用ORDER

BY子句来明确我们的需求。在下面的语句中，结果按照item\_name的字母顺序来排序。

```
SELECT id, item_name, curr_qty FROM grocery_inventory
ORDER BY item_name;
```

执行成功的结果如下所示。

```
+-----+-----+-----+
| id | item_name          | curr_qty |
+-----+-----+-----+
| 1  | Apples             | 1000     |
| 4  | Bottled Water (12-pack) | 500      |
| 3  | Bottled Water (6-pack) | 250      |
| 2  | Bunches of Grapes  | 500      |
+-----+-----+-----+
4 rows in set (0.03 sec)
```

#### 你知道吗？

当从表中查询了结果而没有指定排列顺序的时候，结果可能按照键值排序，也可能不这样排序。由于MySQL重新使用了前面删除的行所占用的空间，就会发生这种情况。换句话说，如果我们添加了ID值从1到5的记录，删除了ID为4的记录，然后又添加了另外一条记录（ID为6），记录可能按照下面的顺序出现在表中。

1, 2, 3, 6, 5.

ORDER BY默认的排序是升序（ASC），字符串顺序是从A到Z，整数顺序是从0开始，日期顺序是从最早的日期到最近的日期。也可以指定一个降序，使用DESC，示例如下。

```
SELECT id, item_name, curr_qty FROM grocery_inventory
ORDER BY item_name DESC;
```

执行的结果如下。

```

+-----+-----+-----+
| id | item_name          | curr_qty |
+-----+-----+-----+
| 2  | Bunches of Grapes  | 500      |
| 3  | Bottled Water (6-pack) | 250      |
| 4  | Bottled Water (12-pack) | 500      |
| 1  | Apples              | 1000     |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

我们并没有受限于只对一个字段排序，可以指定尽可能多的字段，字段之间用逗号隔开。排序优先级按照列出的字段的顺序。

## 16.4.2 限制结果

可以使用LIMIT子句来从SELECT查询结果中返回一定数目的记录。使用LIMIT子句的时候可以有两个参数：偏移量和行数。偏移量是起始位置，而行数应该是自说明的。

假设grocery\_inventory表中有多于两或三条记录，并且我们希望选择按照curr\_qty排序的前两条记录的ID、name和quantity。换句话说，我们希望选取存货最少的两项。如下的单参数限制将会从0位置开始，并且到达第二条记录。

```

SELECT id, item_name, curr_qty FROM grocery_inventory
ORDER BY curr_qty LIMIT 2;

```

执行的结果如下。

```

+-----+-----+-----+
| id | item_name          | curr_qty |
+-----+-----+-----+
| 3  | Bottled Water (6-pack) | 250      |
| 2  | Bunches of Grapes  | 500      |
+-----+-----+-----+
2 rows in set (0.00 sec)

```

LIMIT子句在实际的应用中很有用。例如，我们可以在一组



SELECT语句中使用LIMIT来分步骤遍历结果。首先是头两项，然后是接下来的两项，最后又是接下来的两项，示例如下。

1. SELECT \* FROM grocery\_inventory ORDER BY curr\_qty LIMIT 0, 2;

2. SELECT \* FROM grocery\_inventory ORDER BY curr\_qty LIMIT 2, 2;

3. SELECT \* FROM grocery\_inventory ORDER BY curr\_qty LIMIT 4, 2;

如果在查询中指定了行的一个偏移量和数目，但没有找到结果，我们也不会看到错误，而只是看到一个空的结果集。例如，如果grocery\_inventory表只包含了6条记录，一条带有一个偏移量为6的LIMIT的查询将不会产生结果。

在基于Web的应用程序中，当数据的列表通过一个“前10条”和“后10条”之类的链接来显示的时候，这很可能就需要使用LIMIT子句。

## 16.5 在查询中使用WHERE

我们已经学习了很多方法来从表中获取特定的列，但还没有获取指定的行，而这正是WHERE子句的用武之地。从SELECT语法的例子中，我们看到了WHERE用来指定一个特定的条件。

```
SELECT expressions_and_columns FROM table_name
[WHERE some_condition_is_true]
```

下面的一个例子是，获取那些数量为500的商品的记录。

```
SELECT * FROM grocery_inventory WHERE curr_qty = 500;
```

结果如下所示。

```
+-----+-----+-----+-----+-----+
| id | item_name          | item_desc          | item_price | curr_qty |
+-----+-----+-----+-----+-----+
| 2  | Bunches of Grapes  | Seedless grapes.   | 2.99       | 500      |
| 4  | Bottled Water (12-pack) | 500ml spring water. | 4.49       | 500      |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

正如前面所展示的，如果我们使用一个整数作为WHERE子句的一部分，并不需要引号。需要用引号把字符串括起来，对于转义字符也适用同样的规则，这些规则我们已在前面有关INSERT的小节中学习过。

### 16.5.1 在WHERE子句中使用操作符

我们已经在WHERE子句中使用了相等操作符(=)来确定一个条件的真，也就是说，一个事物是否等于另外一个事物。我们可以使用多种操作符，其中，比较操作符和逻辑操作符是最为常用的类型。表16-1列出了比较操作符及它们的含义。

表16-1 基本比较操作符及其含义

操 作 符	含 义
=	等于
<	小于
!=	不等于
>=	大于或等于
<=	小于或等于
>	大于

还有一个叫做BETWEEN的方便的操作符，它在比较整数或数据的时候很有用，因为它搜索位于一个最小值和最大值之间的结果，示例如下。

```
SELECT * FROM grocery_inventory WHERE item_price BETWEEN 1.50 AND 3.00;
```

结果如下所示。

```
+-----+-----+-----+-----+
| id | item_name          | item_desc          | item_price | curr_qty |
+-----+-----+-----+-----+
| 2  | Bunches of Grapes  | Seedless grapes.   | 2.99       | 500      |
| 3  | Bottled Water (6-pack) | 500ml spring water. | 2.29       | 250      |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

其他的操作符包括逻辑操作符，它们使得我们可以在WHERE子句中使用多个比较。基本逻辑操作符是AND和OR。当使用AND的时候，子句中的所有比较结果必须是真才能得到结果，而使用OR则允许至少有一个比较结果为真。另外，我们可以使用IN操作符来指定想要匹配的商品的一个列表。

## 16.5.2 使用LIKE比较字符串

前面已经介绍了在一个WHERE子句中通过使用=或!=来匹配字符串，但是，还有另外一种有用的操作符可以用来在WHERE子句中比较字符串，这就是LIKE操作符。这个操作符在模式匹配中可以使用如下两个字符作为通配符。

- %——匹配多个字符。
- \_——匹配一个字符。

例如，如果我们要在grocery\_inventory表中找到名字的第一个字母为A的商品，可以使用如下语句。

```
SELECT * FROM grocery_inventory WHERE item_name LIKE 'A%';
```

结果如下所示。

```
+-----+-----+-----+-----+-----+
| id | item_name | item_desc          | item_price | curr_qty |
+-----+-----+-----+-----+-----+
| 1 | Apples   | Beautiful, ripe apples. | 0.25 | 1000 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

提示：

除非在一个二进制字符串上执行一个**LIKE**比较，否则这个比较总是不区分大小写的。我们可以使用**BINARY**关键字来强制执行一个区分大小写的比较。

## 16.6 从多个表中查询

我们并没有受到限制只能同时查询一个表。如果是那样的话，应用程序编程将会是一个漫长而枯燥的任务。当我们在一条SELECT语句中对多个表查询的时候，确实会把这些表连接到一起。

假设我们有两个表：fruit表和color表。我们可以使用两条单独的SELECT语句，分别从这两个表中的每一个来查询行，示例如下。

```
SELECT * FROM fruit;
```

这条查询会产生如下结果。

```
+----+-----+
| id | fruitname |
+----+-----+
| 1  | apple    |
| 2  | orange   |
| 3  | grape    |
| 4  | banana   |
+----+-----+
4 rows in set (0.00 sec)
```

```
SELECT * FROM color;
```

第二条查询会产生如下结果。

```
+----+-----+
| id | colorname |
+----+-----+
| 1  | red       |
| 2  | orange    |
| 3  | purple    |
| 4  | yellow    |
+----+-----+
4 rows in set (0.00 sec)
```

当我们想要一次从两个表查询数据的时候，SELECT语句在语法上

略有不同。首先，必须确保在查询中所使用的所有的表都出现在SELECT语句的FROM子句中。以fruit和color为例，如果只是想从两个表中获取所有的列和行，你可能会考虑使用如下的SELECT语句。

```
SELECT * FROM fruit, color;
```

使用这条查询，我们会得到如下结果。

```
+-----+-----+-----+-----+
| id | fruitname | id | colorname |
+-----+-----+-----+-----+
| 1 | apple     | 1 | red       |
| 2 | orange    | 1 | red       |
| 3 | grape     | 1 | red       |
| 4 | banana    | 1 | red       |
| 1 | apple     | 2 | orange    |
| 2 | orange    | 2 | orange    |
| 3 | grape     | 2 | orange    |
| 4 | banana    | 2 | orange    |
| 1 | apple     | 3 | purple    |
| 2 | orange    | 3 | purple    |
| 3 | grape     | 3 | purple    |
| 4 | banana    | 3 | purple    |
| 1 | apple     | 4 | yellow    |
| 2 | orange    | 4 | yellow    |
| 3 | grape     | 4 | yellow    |
| 4 | banana    | 4 | yellow    |
+-----+-----+-----+-----+
16 rows in set (0.00 sec)
```

16行重复的信息可能不是我们所要查找的。这个查询所做的是把color表中的每一行逐个地连接到fruit表中的每一行。由于fruit表中有4条记录，并且color表中有4条记录，所以返回了16条记录。

当我们想从多个表选择的时候，必须构建正确的WHERE子句来确保确实能够得到想要的记录。在fruit和color表的例子中，我们真正想要的是看到ID相同的来自两个表中的fruitname和colorname记录。这给我们带来了下一个略有不同的查询——当字段在两个表中的名字都相同的时

候，如何表示到底是哪个表中的字段。

很简单，我们只需要给字段名前面添加上表名就行了，如下所示。

`tablename.fieldname`

这样，从两个表中查询ID相等的fruitname和colorname的查询就是如下语句。

```
SELECT fruitname, colorname FROM fruit, color WHERE fruit.id = color.id;
```

这条查询会产生更好一点的结果，示例如下。

```
+-----+-----+
| fruitname | colorname |
+-----+-----+
| apple     | red       |
| orange    | orange    |
| grape     | purple    |
| banana    | yellow    |
+-----+-----+
4 rows in set (0.00 sec)
```

然而，如果我们试图查找在两个表中都出现的且名字相同的一个列，就会得到一个二义性的错误。

```
SELECT id, fruitname, colorname FROM fruit, color
WHERE fruit.id = color.id;
```

这条查询会产生如下的错误。

```
ERROR 1052: Column: 'id' in field list is ambiguous
```

如果是想要从fruit表中获取ID，使用如下语句。

```
SELECT fruit.id, fruitname, colorname FROM fruit,
color WHERE fruit.id = color.id;
```

这条查询会产生如下结果。



```

+-----+-----+-----+
| id    | fruitname | colorname |
+-----+-----+-----+
| 1     | apple    | red       |
| 2     | orange   | orange    |
| 3     | grape    | purple    |
| 4     | banana   | yellow    |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

这是把两个表连接在一起以便在一个单个的SELECT查询中使用的一个基本的例子。JOIN关键字实际上是SQL的一部分，它使得我们能够构建更为复杂的查询。

### 16.6.1 使用JOIN

在MySQL中，有几种类型的JOIN可供使用，所有的这些都涉及到表组合到一起的顺序以及结果显示的顺序。与fruit表和color表一起使用的JOIN叫做INNER JOIN，尽管不必显式地这样写。要使用正确的INNER JOIN语法来重新编写上述SQL语句，示例如下。

```

SELECT fruitname, colorname FROM fruit
INNER JOIN color ON fruit.id = color.id;

```

结果如下所示。

```

+-----+-----+
| fruitname | colorname |
+-----+-----+
| apple    | red       |
| orange   | orange    |
| grape    | purple    |
| banana   | yellow    |
+-----+-----+
4 rows in set (0.00 sec)

```

ON子句替代了我们前面所见到的WHERE子句，在这个例子中，它

告诉MySQL把表中ID相匹配的行连接起来。当使用ON子句连接表的时候，可以使用那些能够在WHERE子句中使用的任何条件，包括所有各种逻辑操作符和算术操作符。

另一种常见的JOIN类型是LEFT JOIN。使用LEFT JOIN把两个表连接到一起的时候，第一个表中的所有行都将返回，不管它在第二个表中是否有匹配。假设在一个地址簿中有两个表，一个叫做master\_name，包含基本的记录，一个叫做email，包括Email记录。email表中的任何记录都将绑定到master\_name表中的一条记录的一个特定ID。首先看看这两个表的内容。

name_id	firstname	lastname
1	John	Smith
2	Jane	Smith
3	Jimbo	Jones
4	Andy	Smith
5	Chris	Jones
6	Anna	Bell
7	Jimmy	Carr
8	Albert	Smith
9	John	Doe

name_id	email
2	jsmith@jsmith.com
6	annabell@aol.com
9	jdoe@yahoo.com

在这两个表上使用LEFT JOIN，我们可以看到，如果email表中的一个值不存在，一个空值将会出现在一个Email地址的位置，执行如下语句。

```
SELECT firstname, lastname, email FROM master_name
LEFT JOIN email ON master_name.name_id = email.name_id;
```

这条LEFT JOIN查询产生如下的结果。

```
+-----+-----+-----+
| firstname | lastname | email |
+-----+-----+-----+
| John      | Smith   |       |
| Jane      | Smith   | jsmith@jsmith.com |
| Jimbo     | Jones   |       |
| Andy      | Smith   |       |
| Chris     | Jones   |       |
| Anna      | Bell    | annabell@aol.com |
| Jimmy     | Carr    |       |
| Albert    | Smith   |       |
| John      | Doe     | jdoe@yahoo.com |
+-----+-----+-----+
9 rows in set (0.00 sec)
```

一个RIGHT JOIN的作用和LEFT JOIN类似，只不过表的顺序相反。换句话说，当使用RIGHT JOIN的时候，第二个表中的所有的行都将返回，不管它们在第一个表中是否有匹配。然而，在master\_name和email表的例子中，email表中只有3行，而master\_name表中有9行。这就意味着，master\_name表中的9行中只有3行会返回，执行如下语句。

```
SELECT firstname, lastname, email FROM master_name
RIGHT JOIN email ON master_name.name_id = email.name_id;
```

结果和预期的一样，如下所示。

```
+-----+-----+-----+
| firstname | lastname | email |
+-----+-----+-----+
| Jane      | Smith   | jsmith@jsmith.com |
| Anna      | Bell    | annabell@aol.com |
| John      | Doe     | jdoe@yahoo.com |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

MySQL中有几种不同类型的JOIN可以使用，我们已经学习了最常见的类型。要学习CROSS JOIN、STRAIGHT JOIN和NATURAL JOIN这样的JOIN，请查看位于<http://dev.mysql.com/doc/refman/5.5/en/join.html>的MySQL手册。当你继续学习的时候，我强烈建议你了解并练习JOIN，这是SQL工具箱中最强大的一款工具之一。

## 16.6.2 使用子查询

简单地说，一个子查询就是出现在另一条SQL语句中的一条SELECT语句。这样的查询很有用，因为它们往往去除了大量的JOIN查询的需要。在应用程序编程的例子中，子查询可以避免了在循环中进行多个查询的必要。

基本子查询语法的一个例子如下所示。

```
SELECT expressions_and_columns FROM table_name WHERE somecolumn = (SUBQUERY);
```

我们还可以用带有UPDATE和DELETE语句的子查询，如下所示。

```
DELETE FROM table_name WHERE somecolumn = (SUBQUERY);
```

或

```
UPDATE table_name SET somecolumn = 'something' WHERE somecolumn = (SUBQUERY);
```

**提示：**

一个子查询的外围语句可以是SELECT、INSERT、UPDATE、DELETE、SET或DO。子查询必须总是出现在括号中，没有例外。

当我们使用子查询的时候，外部语句的WHERE部分不一定必须使

用=比较操作符。除了=，我们可以使用任何的基本比较操作符以及IN这样的关键字。

如下的例子使用一个子查询来获取master\_name表中那些在email表中拥有一个Email地址的用户的记录。

```
SELECT firstname, lastname FROM master_name
WHERE name_id IN (SELECT name_id FROM email);
```

这条查询的结果可能如下所示。

```
+ - - - - - + - - - - - +
| firstname | lastname |
+-----+-----+
| Jane      | Smith    |
| Anna      | Bell     |
| John      | Doe      |
+-----+-----+
3 rows in set (0.00 sec)
```

要了解关于子查询的更多讨论，包括限制，请参考位于<http://dev.mysql.com/doc/refman/5.5/en/subqueries.html> 的MySQL手册的子查询部分。

## 16.7 使用UPDATE命令来修改记录

UPDATE是用来修改已有的单条或多条记录中的一列或多列的内容的SQL命令。最基本的UPDATE语法如下所示。

```
UPDATE table_name
SET column1='new value',
column2='new value2'
[WHERE some_condition_is_true]
```

更新一条记录的规则和插入一条记录的规则类似：输入的数据必须和字段的数据类型对应，并且必须用单引号或双引号把字符串括起来，必要的时候要转义。例如，假设你有一个名为fruit的表，其中包含一个ID、一个水果名称和水果的状态（ripe或rotten）。

```
+-----+-----+-----+
| id | fruit_name | status |
+-----+-----+-----+
| 1 | apple      | ripe   |
| 2 | orange     | rotten |
| 3 | grape      | ripe   |
| 4 | banana     | rotten |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

要把水果的状态改为ripe，使用如下语句。

```
UPDATE fruit SET status = 'ripe';
```

你将会得到来自数据库的如下一条响应。

```
Query OK, 2 rows affected (0.00 sec)
Rows matched: 4  Changed: 2  Warnings: 0
```

进一步看一下这个查询的结果，它成功执行了，我们从Query OK消息就可以看出来。还要注意，只有两行受到影响，如果我们要把一列的

值设置为它已经拥有的值，更新将不会对该列进行。响应消息的第2行显示，有4行已经匹配了，并且只有两行修改了。如果想知道哪些行匹配上了，答案很简单。因为我们没有制定一个特定的条件进行匹配，那么所有的行都匹配上了。

在更新一个表的时候，我们必须小心仔细并且使用一个条件，除非真的想把所有记录的所有列都修改为相同的值。为了证明这一点，假设“grape”在表中没有拼写正确，并且我们想使用UPDATE来修改这个错误。这个查询将会产生可怕的后果。

```
UPDATE fruit SET fruit_name = 'grape';
```

该查询的结果可能很糟糕，如下所示。

```
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0
```

当我们读取结果的时候，将会陷入噩梦中，4条记录都修改了，这意味着fruit表内容如下所示。

```
+----+-----+-----+
| id | fruit_name | status |
+----+-----+-----+
|  1 | grape      | ripe   |
|  2 | grape      | ripe   |
|  3 | grape      | ripe   |
|  4 | grape      | ripe   |
+----+-----+-----+
4 rows in set (0.00 sec)
```

现在，所有的水果记录都是grape。然而，当你试图更改一条记录的拼写的时候，所有的记录都会更改，因为你没有指定一个条件。在将UPDATE权限赋予用户的时候，考虑一下你给予某人的责任，一次错误的更新，整个表都将是grape。在前面的例子中，你本来应该在WHERE

子句中使用id或fruit\_name字段，正如我们在后面的小节将要看到的。

### 16.7.1 条件式UPDATE

进行一次条件式UPDATE意味着使用WHERE子句来匹配特定记录。在UPDATE语句中使用一个WHERE子句和在SELECT语句中使用一个WHERE子句是相同的。所有相同的比较操作符和逻辑操作符一样可以使用，例如等于、大于、OR和AND。

假设fruit表没有完全用grape填充而是包含4条记录，其中的一条记录有一个拼写错误，写成了grappe而不是grape。用来修改拼写错误的UPDATE语句如下所示。

```
UPDATE fruit SET fruit_name = 'grape' WHERE fruit_name = 'grappe';
```

在这个例子中，只有一行匹配，并且只有一行修改，如下面的结果所示。

```
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

我们的fruit表应该是完整的，并且，所有的水果名应该是拼写正确的。

```
SELECT * FROM fruit;
```

这条SELECT查询显示如下结果。



```

+----+-----+-----+
| id | fruit_name | status |
+----+-----+-----+
| 1 | apple      | ripe   |
| 2 | pear       | ripe   |
| 3 | banana     | ripe   |
| 4 | grape      | ripe   |
+----+-----+-----+
4 rows in set (0.00 sec)

```

## 16.7.2 在UPDATE中使用已有的列值

UPDATE的另一个功能是使用记录中当前的值作为一个基准值。例如，回到grocery\_inventory表的例子，假设有如下的一张表。

```

+----+-----+-----+-----+-----+
| id | item_name          | item_desc          | item_price | curr_qty |
+----+-----+-----+-----+-----+
| 1 | Apples             | Beautiful, ripe apples. | 0.25      | 1000     |
| 2 | Bunches of Grapes  | Seedless grapes.      | 2.99      | 500      |
| 3 | Bottled Water (6-pack) | 500ml spring water.  | 2.29      | 250      |
| 4 | Bottled Water (12-pack) | 500ml spring water.  | 4.49      | 500      |
| 5 | Bananas            | Bunches, green.       | 1.99      | 150      |
| 6 | Pears              | Anjou, nice and sweet. | 0.5       | 500      |
| 7 | Avocado            | Large Haas variety.   | 0.99      | 750      |
+----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

当某个人购买了一件商品，例如一个苹果（id=1），inventory表将相应地更新。然而，我们不知道在curr\_qty列中输入什么数值，只知道卖掉了一件。在这种情况下，使用该列的当前值并且减1，语句如下。

```
UPDATE grocery_inventory SET curr_qty = curr_qty - 1 WHERE id = 1;
```

这会在curr\_qty列给出一个999的新值，实际上正是如此。

```
SELECT * FROM grocery_inventory;
```

这条SELECT查询显示了新的库存数量，如下所示。

id	item_name	item_desc	item_price	curr_qty
1	Apples	Beautiful, ripe apples.	0.25	999
2	Bunches of Grapes	Seedless grapes.	2.99	500
3	Bottled Water (6-pack)	500ml spring water.	2.29	250
4	Bottled Water (12-pack)	500ml spring water.	4.49	500
5	Bananas	Bunches, green.	1.99	150
6	Pears	Anjou, nice and sweet.	0.5	500
7	Avocado	Large Haas variety.	0.99	750

7 rows in set (0.00 sec)

## 16.8 使用REPLACE命令

修改记录的另一个方法是使用REPLACE命令，它类似于INSERT命令。

```
REPLACE INTO table_name (column list) VALUES (column values);
```

REPLACE语句像这样工作：如果插入到表中的记录包含了一个主键值，这个主键值和表中已有的一条记录的主键值相等，表中的记录将被删除掉，并且新的记录会插入到它所在的位置。

提示：

REPLACE命令是MySQL对ANSI SQL的一个特定的扩展。这条命令模拟了DELETE的操作并重新INSERT一条特定记录的操作。换句话说，我们用一条命令完成了两条命令的工作。

使用grocery\_inventory表，如下的命令将替换Apple的记录。

```
REPLACE INTO grocery_inventory VALUES  
(1, 'Granny Smith Apples', 'Sweet!', '0.50', 1000);
```

结果如下所示。

```
Query OK, 2 rows affected (0.00 sec)
```

在查询结果中，注意结果状态中的“2 rows affected”。在这个例子中，由于id是一个在grocery\_inventory表中有对应的值的一个主键，最初的行被删除而一个新行插入，因此是2行受到影响。使用SELECT语句查询数据以验证记录是正确的，结果如下。

```

+-----+-----+-----+-----+-----+
| id | item_name          | item_desc          | item_price | curr_qty |
+-----+-----+-----+-----+-----+
| 1 | Granny Smith Apples | Sweet!             | 0.50       | 1000     |
| 2 | Bunches of Grapes  | Seedless grapes.   | 2.99       | 500      |
| 3 | Bottled Water (6-pack) | 500ml spring water. | 2.29       | 250      |
| 4 | Bottled Water (12-pack) | 500ml spring water. | 4.49       | 500      |
| 5 | Bananas             | Bunches, green.    | 1.99       | 150      |
| 6 | Pears               | Anjou, nice and sweet. | 0.5        | 500      |
| 7 | Avocado             | Large Haas variety. | 0.99       | 750      |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

如果使用一条REPLACE语句，并且新记录中的主键的值没有和表中已经存在的一个主键的值匹配，这条记录会直接插入，这样就只有一条记录受到影响。

## 16.9 使用DELETE命令

DELETE的基本语法如下。

```
DELETE FROM table_name
[WHERE some_condition_is_true]
[LIMIT rows]
```

注意，在DELETE命令中如果没有条件，那么当你使用DELETE的时候，表中所有记录都会被删除掉。你可能还记得在本章前面的一次关于fruit表中的grape的失败，当更新一个表而没有指定条件的时候，导致所有的记录都更新了。在使用DELETE时候也要小心出现类似的情况。

假设一个fruit的表的结构和数据如下所示。

```
+-----+-----+-----+
| id | fruit_name | status |
+-----+-----+-----+
| 1 | apple      | ripe   |
| 2 | pear       | rotten |
| 3 | banana     | ripe   |
| 4 | grape      | rotten |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

如下的语句删除了该表中的所有记录。

```
DELETE FROM fruit;
```

我们总是可以通过对表使用SELECT语句来验证删除。在删除所有记录后执行下面这条命令。

```
SELECT * FROM fruit;
```

将会看到fruit表中所有的水果都已经删除掉了。

Empty set (0.00 sec)

## 条件式DELETE

一个条件式DELETE语句，和条件式SELECT或UPDATE语句一样，意味着我们使用WHERE子句来匹配特定的记录。我们有了全部可用的比较操作符和逻辑操作符，因此，可以挑选和选取要删除哪些记录。

一个基本的例子是，从fruit表中删除所有状态为rotten的水果的记录，如下所示。

```
DELETE FROM fruit WHERE status = 'rotten';
```

将会删除两条记录，而只有成熟的水果会保留下来。

Query OK, 2 rows affected (0.00 sec)

查询的结果如下。

```
+-----+-----+-----+
| id | fruit_name | status |
+-----+-----+-----+
| 1 | apple      | ripe   |
| 3 | banana     | ripe   |
+-----+-----+-----+
2 rows in set (0.00 sec)
```

我们也可以在DELETE语句中使用ORDER BY子句，下面看一下把ORDER BY子句添加到结构中以后的基本DELETE语法。

```
DELETE FROM table_name
[WHERE some_condition_is_true]
[ORDER BY some_column [ASC | DESC]]
[LIMIT rows]
```

乍一看，你可能会问：“为什么我删除记录与顺序有关系呢”？

ORDER BY子句不是用来删除记录的，而是用来排序记录的。

在这个例子中，一个名为access\_log的表给出了访问时间和用户名，如下所示。

```
+-----+-----+-----+
| id | date_accessed | username |
+-----+-----+-----+
| 1 | 2012-01-06 06:09:13 | johndoe |
| 2 | 2012-01-06 06:09:22 | janedoe |
| 3 | 2012-01-06 06:09:39 | jsmith  |
| 4 | 2012-01-06 06:09:44 | mikew   |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

要删除最早的记录，首先使用ORDER BY来正确地排序记录，然后，使用LIMIT来仅删除一条记录，如下所示。

```
DELETE FROM access_log ORDER BY date_accessed DESC LIMIT 1;
```

查询access\_log的记录，从而验证只有如下3条记录存在。

```
SELECT * FROM access_log;
```

结果如下所示。

```
+-----+-----+-----+
| id | date_accessed | username |
+-----+-----+-----+
| 2 | 2012-01-06 06:09:22 | janedoe |
| 3 | 2012-01-06 06:09:39 | jsmith  |
| 4 | 2012-01-06 06:09:44 | mikew   |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

## 16.10 MySQL中常用的字符串函数

MySQL内建的和字符串相关的函数，可以以几种方式使用。我们可以在SELECT语句中使用函数而不用指定一个表就可以获取函数的结果。或者我们可以通过把两个字段连接为一个新的字符串，使用函数来扩展SELECT的结果。后面的例子决不是MySQL与字符串相关的函数的完整的库。要了解更多信息，请参考位于

<http://dev.mysql.com/doc/refman/5.5/en/string-functions.html> 的MySQL 手册。

### 16.10.1 长度和连接函数

长度和连接函数主要用于字符串的长度和把字符串连接起来。与长度相关的函数包括LENGTH()、OCTET\_LENGTH()、CHAR\_LENGTH()和CHARACTER\_LENGTH()，它们所做的事情实际上是相同的，即计算字符串中的字符数。

```
SELECT LENGTH('This is cool!');
```

结果如下所示。

```
+-----+
| LENGTH('This is cool!') |
+-----+
|                        13 |
+-----+
1 row in set (0.00 sec)
```

有趣的事情从CONCAT()函数开始，这个函数用来把两个或多个字符串连接起来。



```
SELECT CONCAT('My', 'S', 'QL');
```

这条查询的结果如下所示。

```
+-----+
| CONCAT('My', 'S', 'QL') |
+-----+
| MySQL                    |
+-----+
1 row in set (0.00 sec)
```

假设我们对一个包含了名字的表使用这个函数，名字划分为 `firstname` 字段和 `lastname` 字段。我们使用两个字段名来连接 `firstname` 字段和 `lastname` 字段的值，而不是使用两个字符串。通过连接字段，减少了在应用程序中得到相同结果所需的代码。

```
SELECT CONCAT(firstname, lastname) FROM master_name;
```

如果如下。

```
+-----+
| CONCAT(firstname, lastname) |
+-----+
| JohnSmith                   |
| JaneSmith                   |
| JimboJones                  |
| AndySmith                   |
| ChrisJones                  |
| AnnaBell                    |
| JimmyCarr                   |
| AlbertSmith                 |
| JohnDoe                     |
+-----+
9 rows in set (0.00 sec)
```

你知道吗？

如果在函数中使用一个字段名而不是一个字符串，不要把字段名包含到引号中。如果你这么做了，MySQL逐字地解析字符串。在CONCAT()的例子中，我们得到如下结果。

```
SELECT CONCAT('firstname', 'lastname') FROM master_name;
```

[illegible]

如果名字之间有某种类型的分隔符，CONCAT()函数将会很有用，这也就引出了下一个函数。

你可能还记得，`CONCAT_WS()`代表使用分隔符的连接。可以选择任何分隔符，下面的例子使用空格。

```
SELECT CONCAT WS(' ', firstname, lastname) FROM master name;
```

这条查询的结果如下所示。

```
+-----+
| CONCAT_WS(' ', firstname, lastname) |
+-----+
| John Smith |
| Jane Smith |
| Jimbo Jones |
| Andy Smith |
| Chris Jones |
| Anna Bell |
| Jimmy Carr |
| Albert Smith |
| John Doe |
+-----+
9 rows in set (0.00 sec)
```

如果想要缩短结果表的宽度，可以使用AS来命名自定义结果字段，示例如下。

```
SELECT CONCAT_WS(' ', firstname, lastname) AS fullname FROM master_name;
```

使用这条语句，将会得到如下结果。

```
+-----+
| fullname |
+-----+
| John Smith |
| Jane Smith |
| Jimbo Jones |
| Andy Smith |
| Chris Jones |
| Anna Bell |
| Jimmy Carr |
| Albert Smith |
| John Doe |
+-----+
9 rows in set (0.00 sec)
```

### 16.10.2 截断和填充函数

MySQL提供了几个函数来向字符串中添加和删除额外的字符，包括空格。RTRIM()和LTRIM()函数从一个字符串的右端或者左端删除空格。

```
SELECT RTRIM('stringstring ');
```

该查询的结果如下所示，尽管很难看出它的变化。

```
+-----+
| RTRIM('stringstring  ') |
+-----+
| stringstring |
+-----+
1 row in set (0.00 sec)
```

LTRIM()函数使得更容易看明白，示例如下。

```
SELECT LTRIM('  stringstring');
```

这条查询导致如下结果，空白明显被去掉了，示例如下。

```
+-----+
| LTRIM('  stringstring') |
+-----+
| stringstring             |
+-----+
1 row in set (0.00 sec)
```

如果字符串产生自一个固定宽度的字段，并且它要么不需要添加其他的填充，要么将插入到一个varchar或其他非固定宽度的字段，我们可以截断填充过的字符串。如果字符串使用空格以外的其他字符来填充，使用TRIM()函数来指定想要删除的字符。例如，要把前面的X字符从字符串XXXneedleXXX中删除，使用如下语句。

```
SELECT TRIM(LEADING 'X' FROM 'XXXneedleXXX');
```

这条查询的结果如下。

[illegible]

可以使用TRAILING从字符串末尾删除字符。

```
SELECT TRIM(TRAILING 'X' FROM 'XXXneedleXXX');
```

这条查询的结果如下所示。

[illegible]

如果没有指定LEADING或TRAILING，则假设两种方式都使用，语句如下。

```
SELECT TRIM('X' FROM 'XXXneedleXXX');
```

这条查询结果如下。

```
+-----+
| TRIM('X' FROM 'XXXneedleXXX') |
+-----+
| needle                          |
+-----+
1 row in set (0.00 sec)
```

和RTRIM()和LTRIM()删除填充字符一样，RPAD()和LPAD()向一个字符串添加字符。例如，在用于销售的一个数据库中，对于作为订单号的一部分的一个字符串，我们可能想要添加一个特定的标识字符。当我们使用填充函数时，所需的元素是字符串、目标长度以及填充字符。例如，用一个X字符填充字符串needle，直到字符串达到10个字符的长度，使用如下语句。

```
SELECT RPAD('needle', 10, 'X');
```

结果如下。

```
+-----+
| RPAD('needle', 10, 'X') |
+-----+
| needleXXXXX             |
+-----+
1 row in set (0.00 sec)
```

### 16.10.3 定位和位置函数

定位和位置函数用来在另一个字符串中查找一个字符串的部分。LOCATE()函数返回一个给定的子字符串在目标字符串中第一次出现的

位置。例如，我们可以在一个haystack中查找needle，语句如下。

```
SELECT LOCATE('needle', 'haystackneedlehaystack');
```

应该会看到如下结果。

```
+-----+
| LOCATE('needle', 'haystackneedlehaystack') |
+-----+
|                                           9 |
+-----+
1 row in set (0.00 sec)
```

子字符串needle从目标字符串的第9个字符开始。如果没有在目标字符串中找到子字符串，MySQL返回0作为结果。

提示：

和大多数程序设计语言中的位置计算不同，那些语言中的位置计算都是从0开始的，而MySQL则是从1开始计算位置的。

LOCATE()函数的一个扩展是对起始位置使用第3个参数。如果从haystack中的位置9之前开始查找needle，我们将能够得到一个9的结果。否则，由于needle是从位置9开始的，如果我们指定了一个更大的数字作为开始位置，将会得到一个0的结果。

#### 16.10.4 子字符串函数

从一个目标字符串中提取一个子字符串，有几个函数能够满足要求。给定一个字符串、起始位置和长度，我们可以使用SUBSTRING()函数。如下例子从字符串MySQL中获取3个字符，从位置2开始。

```
SELECT SUBSTRING("MySQL", 2, 3);
```

结果如下所示。

```
+-----+
| SUBSTRING("MySQL", 2, 3) |
+-----+
| ySQ                      |
+-----+
1 row in set (0.00 sec)
```

如果只想要字符串左端或右端的几个字符，可以使用LEFT()和RIGHT()函数，示例如下。

```
SELECT LEFT("MySQL", 2);
```

这条查询的结果如下。

```
+-----+
| LEFT("MySQL", 2) |
+-----+
| My               |
+-----+
1 row in set (0.00 sec)
```

同样的，使用RIGHT()函数示例如下。

```
SELECT RIGHT("MySQL", 3);
```

这条查询的结果如下。

```
+-----+
| RIGHT("MySQL", 3) |
+-----+
| SQL               |
+-----+
1 row in set (0.00 sec)
```

子字符串函数的很多常见用法之一就是提取订单号码的一部分来看看是谁下了这个订单。在一些应用程序中，系统设计来产生一个包含日期、客户身份和其他信息的订单号码。如果这个订单号码总是遵守一种特定的模式，如XXXX-YYYYYY-ZZ，我们可以使用子字符串函数来提

取整个订单号码的单个部分。例如，如果ZZ总是表示订单要发往哪个州，我们可以使用RIGHT()函数来提取这些字符并且显示订单中关于送往哪个州的数字。

### 16.10.5 字符串修改函数

我们所选择的程序设计语言可能有修改字符串的函数，但是，如果可以执行任务作为SQL语句的一部分，那会更好，尽可能地让数据库系统做更多的工作。

MySQL的LCASE()和UCASE()函数把一个字符串转换为小写的或大写的，示例如下。

```
SELECT LCASE('MYSQL');
```

这条查询的结果如下

```
+-----+
| LCASE('MYSQL') |
+-----+
| mysql          |
+-----+
1 row in set (0.00 sec)
```

要将其变成大写字符，使用如下语句。

```
SELECT UCASE('mysql');
```

这条查询将会得到如下结果。

```
+-----+
| UCASE('mysql') |
+-----+
| MYSQL          |
+-----+
1 row in set (0.00 sec)
```



**提示：**

当我们根据存储在MySQL中的数据验证用户输入的时候，例如，在用户登录表单的情况下，LCASE()和UCASE()函数很实用。如果我们想要登录过程不区分大小写，可以尝试用用户输入的大写（或小写）版本来匹配存储在表中的数据的大写（或小写）版本。

记住，如果把字段名和函数一起使用，不要使用引号，示例如下。

```
SELECT UCASE(lastname) FROM master_name;
```

使用上面的查询，将会得到如下的结果。

```
+-----+
| UCASE(lastname) |
+-----+
| BELL            |
| CARR            |
| DOE             |
| JONES           |
| JONES           |
| SMITH           |
| SMITH           |
| SMITH           |
| SMITH           |
+-----+
9 rows in set (0.00 sec)
```

另一个有趣的字符串操作函数是REPEAT()，它所做的事情正如其名字那样，重复一个字符串给定的次数，示例如下。

```
SELECT REPEAT("bowwow", 4);
```

应该会看到如下的结果。

```
+-----+
| REPEAT("bowwow", 4) |
+-----+
| bowwowbowwowbowwowbowwow |
+-----+
1 row in set (0.00 sec)
```

REPLACE()函数把另一个字符串中一个给定字符串的所有出现都替换掉，示例如下。

```
SELECT REPLACE('bowwowbowwowbowwowbowwow', 'wow', 'WOW');
```

这条查询将会产生如下的结果。

```
+-----+
| REPLACE( 'bowwowbowwowbowwowbowwow', 'wow', 'WOW') |
+-----+
| bowWOWbowWOWbowWOWbowWOW |
+-----+
1 row in set (0.00 sec)
```

## 16.11 在MySQL中使用日期和时间函数

MySQL内建的和日期相关的函数可以用在SELECT语句中来获取函数的结果，指定或不指定一个表都可以。或者，可以把任何类型的日期字段（如日期、日期时间、时间戳、年份等）和函数一起使用。

根据所用的字段类型，和日期相关的函数的结果或多或少都有用处。下面的例子绝不是MySQL日期和时间函数的完整库。要了解更多信息，请参阅位于<http://dev.mysql.com/doc/refman/5.0/en/date-and-time-functions.html> 的MySQL手册。

### 16.11.1 操作日期

DAYOFWEEK()和WEEKDAY()函数做类似的事情，但是结果略有不同。这两个函数都用来查找一个日期的星期索引，但是，不同之处在于开始日期和位置。

如果使用DAYOFWEEK(), 一个星期的第一天是星期日, 位置为1。一个星期的最后一天是星期六, 位置为7, 示例如下。

```
SELECT DAYOFWEEK( '2012-01-09' );
```

这条查询产生如下的结果。

```
+-----+
| DAYOFWEEK('2012-01-09') |
+-----+
|                2 |
+-----+
1 row in set (0.00 sec)
```

结果显示，2006年1月9日的星期索引是2，也就是说该天是星期一。对于WEEKDAY()函数使用同样的日期，将会得到不同的结果，但含义是相同的，示例如下。

```
+-----+
| WEEKDAY('2012-01-09') |
+-----+
|                        0 |
+-----+
1 row in set (0.00 sec)
```

结果显示，2006年1月9日的星期索引是0。因为WEEKDAY()使用星期一作为星期的第一天，位置为0，而使用星期日作为最后一天，位置为6。结果为0是正确的，表示这一天是星期一。

DAYOFMONTH()和DAYOFYEAR()函数更加直接，只有一个结果，并且，DAYOFMONTH()的结果的范围从1到31，而DAYOFYEAR()的范围从1到366。看下面的例子。

```
SELECT DAYOFMONTH('2012-01-09');
```

这条查询产生如下的结果。

```
+-----+
| DAYOFMONTH('2012-01-09') |
+-----+
|                        9 |
+-----+
1 row in set (0.00 sec)
```

现在试试下面的例子。

```
SELECT DAYOFYEAR('2012-01-09');
```

这条查询产生如下的结果。

```

+-----+
| DAYOFYEAR('2012-01-09') |
+-----+
|                09 |
+-----+
1 row in set (0.00 sec)

```

根据一个特定的日期来返回它在月份中的天数，看上去似乎有些奇怪，因为这个天数就在日期字符串中。但是，考虑一下在WHERE子句中使用这一函数来对记录进行比较。如果我们有一个表保存了在线订单，其中一个字段包含了下订单的日期，我们可以很快地得到一周中任何一天的订单数量，或者看到前半个月和后半个月分别有多少订单。

如下的两个查询显示了在每个星期的前三天中（包括各个月）以及这周其他的各天中的订单数。

```
SELECT COUNT(id) FROM orders WHERE DAYOFWEEK(date_ordered) < 4;
```

```
SELECT COUNT(id) FROM orders WHERE DAYOFWEEK(date_ordered) > 3;
```

使用DAYOFMONTH(), 如下的例子给出了在任何一个月份中前半个月和后半月的订单数目。

```
SELECT COUNT(id) FROM orders WHERE DAYOFMONTH(date_ordered) < 16;
```

```
SELECT COUNT(id) FROM orders WHERE DAYOFMONTH(date_ordered) > 15;
```

可以使用DAYNAME()函数来为结果添加更多的生机，因为它会返回给定日期的星期名。

```
SELECT DAYNAME(date_ordered) FROM orders;
```

这条查询产生如下的结果。

```

+-----+
| DAYNAME(date_ordered) |
+-----+
| Thursday              |
| Monday                |
| Thursday              |
| Thursday              |
| Wednesday             |
| Thursday              |
| Sunday                |
| Sunday                |
+-----+
8 rows in set (0.00 sec)

```

函数并不仅限于用在WHERE子句中，也可以在ORDER BY子句中使用它们，示例如下。

```
SELECT DAYNAME(date_ordered) FROM orders ORDER BY DAYOFWEEK(date_ordered);
```

### 16.11.2 操作月份和年份

星期几并不是日历的唯一部分，MySQL也有专门用于月份和年份的函数。就像DAYOFWEEK()和DAYNAME()函数一样，MONTH()和MONTHNAME()函数返回了一年中的月份数和给定日期的月份的名字，示例如下。

```
SELECT MONTH('2012-01-09'), MONTHNAME('2012-01-09');
```

这条查询产生如下的结果。

```

+-----+-----+
| MONTH('2012-01-09') | MONTHNAME('2012-01-09') |
+-----+-----+
| 1                   | January                  |
+-----+-----+
1 row in set (0.00 sec)

```

orders表使用MONTHNAME()来显示相应的结果，但很多都是重复的数据，示例如下。

```

+-----+
| MONTHNAME(date_ordered) |
+-----+
| November                |
| November                |
| November                |
| November                |
| November                |
| November                |
| November                |
| November                |
| October                 |
+-----+
8 rows in set (0.00 sec)

```

可以使用DISTINCT来获取非重复性的结果，示例如下。

```
SELECT DISTINCT MONTHNAME(date_ordered) FROM orders;
```

这条查询产生如下的结果。

```

+-----+
| MONTHNAME(date_ordered) |
+-----+
| November                |
| October                 |
+-----+
2 rows in set (0.00 sec)

```

要操作年份，YEAR()函数返回给定日期的年份，示例如下。

```
SELECT DISTINCT YEAR(date_ordered) FROM orders;
```

这条查询产生如下的结果。

```

+-----+
| YEAR(date_ordered)      |
+-----+
| 2011                    |
| 2012                    |
+-----+
1 row in set (0.00 sec)

```

### 16.11.3 操作周

操作周可能是有些技巧性的地方，如果星期日是一周的第一天而12月份还没有到达一周的结束，那么，一年中就会有53周。例如，2001年的12月30号是星期日。

```
SELECT DAYNAME('2001-12-30');
```

这条查询产生的结果如下。

```
+-----+
| DAYNAME('2001-12-30') |
+-----+
| Sunday                  |
+-----+
1 row in set (0.00 sec)
```

这一事实使得2001年的12月30号是该年份的第53周的一部分。

```
SELECT WEEK('2001-12-30');
```

结果中显示出了该年份所拥有的正确的星期数。

```
+-----+
| WEEK('2001-12-30') |
+-----+
|                    53 |
+-----+
1 row in set (0.00 sec)
```

第53周包含12月30日和12月31日，只有两天，2002年的第一周从1月1日开始。

如果我们想要一个周从星期一开始但仍然能够得到一个年份中的星期数，可选的第二个参数使得我们能够更改开始日期。1表示这个周从星期一开始。在后面的例子中，从星期一开始使得12月30日成为2001年第52周的一部分，但12月31日仍然是2001年第53周的一部分。

```
SELECT WEEK('2001-12-30',1);
```



这条查询产生如下的结果。

```
+-----+
| WEEK('2001-12-30',1) |
+-----+
|                    52 |
+-----+
1 row in set (0.00 sec)
```

而如下的查询。

```
SELECT WEEK('2001-12-31',1);
```

产生如下的结果。

```
+-----+
| WEEK('2001-12-31',1) |
+-----+
|                    53 |
+-----+
1 row in set (0.00 sec)
```

#### 16.11.4 操作小时、分钟和秒

如果我们想使用一个包含了确切时间的日期，例如日期时间或时间戳，或者仅仅是一个时间字段，也有函数能够从字符串中得出小时、分钟和秒。这并不令人惊讶，这些函数叫做HOUR()、MINUTE()和SECOND()。HOUR()返回了给定时间的小时，这个值在0到23之间。MINUTE()和SECOND()的范围在0到59之间。

示例如下。

```
SELECT HOUR('2012-01-09 07:27:49') AS hour,  
MINUTE('2012-01-09 07:27:49') AS minute,  
SECOND('2012-01-09 07:27:49') AS second;
```

这条查询将产生如下的结果。

```
+-----+-----+-----+
| hour | minute | second |
+-----+-----+-----+
|    7 |    27 |    49 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

有很多的查询可以从一个日期时间字段获取一个时间，我们可以把小时和分钟放在一起，甚至可以使用CONCAT\_WS()在结果之间放置一个“:”，从而得到一个时间的表示，示例如下。

```
SELECT CONCAT_WS(':',HOUR('2012-01-09 07:27:49'),
MINUTE('2012-01-09 07:27:49')) AS sample_time;
```

这条查询产生如下的结果。

```
+-----+
| sample_time |
+-----+
| 7:27        |
+-----+
1 row in set (0.00 sec)
```

16.11.5 使用MySQL格式化日期和时间

DATE\_FORMAT()函数把一个日期、日期时间或者时间戳字段格式化为一个字符串，它通过选项来明确制定如何显示结果。

DATE\_FORMAT()的语法如下。

```
DATE_FORMAT(date,format)
```

表16-2列出了很多格式化选项。

表16-2 DATE\_FORMAT()格式化字符串选项

选 项	结 果

%M	月份名称（January到December）
%b	缩略的月份名称（Jan到Dec）
%m	带有填充数字的月份（01到12）
%c	月份（1到12）
%W	星期几的名称（Sunday到Saturday）
%a	缩略的星期几的名称（Sun到Sat）
%D	使用英文后缀的月份中的第几天，如first、second、third等等
%d	带有填充的月份中的第几天（00到31）
%e	月份中的第几天（0到31）
%j	带有填充的年份中的第几天（001到366）
%Y	四位数的年份
%y	两位数的年份
%X	星期日是第一天的四位年份，和%V一起使用
%x	星期一是第一天的四位年份，和%V一起使用

%w	星期几（0=Sunday，...，6=Saturday）
%U	星期日是第一天的星期数（0到53）
%u	星期一是第一天的星期数（0到53）
%V	星期日是第一天的星期数（1到53），和%X一起使用
%v	星期一是第一天的星期数（1到53），和%x一起使用
%H	带有填充位的小时（00到23）
%k	小时（0到23）
%h	带有填充位的小时（01到12）
%l	小时（1到12）
%i	带有填充位的分钟（00到59）
%S	带有填充位的秒数（00到59）
%s	带有填充位的秒数（00到59）
%r	12小时时钟的时间（hh:mm:ss [AP]M）

%T	24小时时钟的时间（hh:mm:ss）
%p	AM或PM

提示：

DATE\_FORMAT()字符串选项中使用的任何其他的字符都是依次出现的。

要显示我们上一小节中提到的02:02的结果，应该使用%h和%i选项来从日期返回小时和分钟，并且在两个选项之间带有一个“:”，示例如下。

```
SELECT DATE_FORMAT('2012-01-09 02:02:00', '%h:%i') AS sample_time;
```

这条查询产生如下的结果。

```
+-----+
| sample_time |
+-----+
| 02:02      |
+-----+
1 row in set (0.00 sec)
```

下面只是几个关于使用DATE\_FORMAT()函数的例子，但是，自己练习使用这个函数才是理解它的最好办法。

```
SELECT DATE_FORMAT('2012-01-09', '%W, %M %D, %Y') AS sample_time;
```

这条查询产生如下的输出。

```

+-----+
| sample_time |
+-----+
| Monday, January 9th, 2012 |
+-----+
1 row in set (0.00 sec)

```

如下是格式化当前时间的一条查询（对了，NOW()表示的是我编写这条语句时候的时间）。

```

SELECT DATE_FORMAT(NOW(), '%W the %D of %M, %Y  
around %l o'clock %p') AS sample_time;

```

这条查询产生如下的输出。

```

+-----+
| sample_time |
+-----+
| Tuesday the 10th of January, 2012 around 8 o'clock PM |
+-----+
1 row in set (0.04 sec)

```

花些时间自己研究日期格式化选项；其功能很完备，并且，你会发现它们很容易使用。

### 16.11.6 使用MySQL执行日期算术

MySQL有几个函数可以用来执行日期算术，这可能是MySQL做计算比使用PHP脚本计算更快的领域之一。给定一个起始日期和一个间隔，DATE\_ADD()和DATE\_SUB()函数返回结果。两个函数的语法分别如下。

```
DATE_ADD(date, INTERVAL value type)
```

```
DATE_SUB(date, INTERVAL value type)
```

表16-3显示了可能的类型及它们期待的值格式。

表16-3 日期算术的值和类型

值	类 型
秒数	SECOND
分钟数	MINUTE
小时数	HOUR
天数	DAY
月份数	MONTH
年数	YEAR
“分钟：秒数”	MINUTE_SECOND
“小时：分钟”	HOUR_4MINUTE
“日期小时”	DAY_HOUR
“年份-月份”	YEAR_MONTH
“小时：分钟：秒数”	HOUR_SECOND
“日期小时：分钟”	DAY_MINUTE

“日期小时：分钟：秒数”	DAY_SECOND
--------------	------------

例如，要得到当前日期加21天的结果，使用如下方法。

```
SELECT DATE_ADD(NOW(), INTERVAL 21 DAY);
```

这条查询产生如下的结果。

```
+-----+
| DATE_ADD(NOW(), INTERVAL 21 DAY) |
+-----+
| 2012-01-31 21:02:16                |
+-----+
1 row in set (0.02 sec)
```

使用DATE\_SUB()产生如下的结果。

```
+-----+
| DATE_SUB(NOW(), INTERVAL 21 DAY) |
+-----+
| 2011-12-20 21:02:23                |
+-----+
1 row in set (0.00 sec)
```

使用如表16-3所示的表达式，不管自然趋势是什么，使用DAY而不是DAYS。使用DAYS会导致一个如下的错误。

```
ERROR 1064: You have an error in your SQL syntax near 'DAYS)' at line 1
```

如果我们对一个日期值而不是日期时间值使用DATE\_ADD()或DATE\_SUB()函数，结果将会显示为一个日期值，除非你使用了关系到小时、分钟和秒钟的表达式。在那种情况下，结果将是一个日期时间。

例如，第一个查询的结果仍然为一个日期字段，而第二个查询的结果变成了一个日期时间。



```
SELECT DATE_ADD("2011-12-31", INTERVAL 1 DAY);
```

这条查询产生如下的结果。

```
+-----+
| DATE_ADD("2011-12-31", INTERVAL 1 DAY) |
+-----+
| 2012-01-01                               |
+-----+
1 row in set (0.00 sec)
```

而执行如下这条查询。

```
SELECT DATE_ADD("2011-12-31", INTERVAL 12 HOUR);
```

产生如下的结果。

```
+-----+
| DATE_ADD("2011-12-31", INTERVAL 12 HOUR) |
+-----+
| 2011-12-31 12:00:00                       |
+-----+
1 row in set (0.00 sec)
```

也可以使用+或-操作符，而不是DATE\_ADD()和DATE\_SUB()函数来执行日期算术，示例如下。

```
SELECT "2011-12-31" + INTERVAL 1 DAY;
```

这条查询产生如下的结果。

```
+-----+
| "2011-12-31" + INTERVAL 1 DAY |
+-----+
| 2012-01-01                               |
+-----+
1 row in set (0.00 sec)
```

### 16.11.7 特殊函数和转换函数

MySQL的NOW()函数返回一个当前日期时间结果，并且对于时间

戳登录或访问时间以及很多其他的任务都很有用。MySQL还有一些其他的函数可以执行类似的任务。

CURDATE()和CURRENT\_DATE()函数是相同的，并且它们都以YYYY-MM-DD格式返回当前日期，示例如下。

```
SELECT CURDATE(), CURRENT_DATE();
```

这条查询产生如下的结果。

```
+-----+-----+
| CURDATE() | CURRENT_DATE() |
+-----+-----+
| 2012-01-10 | 2012-01-10      |
+-----+-----+
1 row in set (0.01 sec)
```

类似地，CURTIME()和CURRENT\_TIME()函数以HH:MM:SS格式返回当前时间，示例如下。

```
SELECT CURTIME(), CURRENT_TIME();
```

这条查询产生如下的结果。

```
+-----+-----+
| CURTIME() | CURRENT_TIME() |
+-----+-----+
| 09:14:26  | 09:14:26       |
+-----+-----+
1 row in set (0.00 sec)
```

NOW()、SYSDATE()和CURRENT\_TIMESTAMP()以完整的日期时间格式（YYYY-MM-DD HH:MM:SS）返回值，示例如下。

```
SELECT NOW(), SYSDATE(), CURRENT_TIMESTAMP();
```

这条查询产生如下的结果。

```

+-----+-----+-----+
| NOW()          | SYSDATE()          | CURRENT_TIMESTAMP() |
+-----+-----+-----+
| 2012-01-10 15:23:52 | 2012-01-10 15:23:52 | 2012-01-10 15:23:52 |
+-----+-----+-----+
1 row in set (0.00 sec)

```

UNIX\_TIMESTAMP()函数以UNIX时间戳的格式返回当前日期，或者把一个给定日期转换为UNIX时间戳的格式。UNIX时间戳的格式是从UNIX时间戳（即1970年1月1日午夜）开始的秒数来表示的，示例如下。

```
SELECT UNIX_TIMESTAMP();
```

该查询运行的时候，产生如下所示的结果。

```

+-----+
| UNIX_TIMESTAMP() |
+-----+
|          1326247953 |
+-----+
1 row in set (0.00 sec)

```

下面的查询获取指定日期的UNIX时间戳。

```
SELECT UNIX_TIMESTAMP('1973-12-30');
```

这条查询的结果如下所示。

```

+-----+
| UNIX_TIMESTAMP('1973-12-30') |
+-----+
|                  126086400 |
+-----+
1 row in set (0.00 sec)

```

当没有任何选项的时候，FROM\_UNIXTIME()函数执行从一个UNIX时间戳到一个完整的日期时间格式的转换，示例如下。

```
SELECT FROM_UNIXTIME('1326247953');
```

我们可以使用DATE\_FORMAT()函数的格式选项来以更为整齐的方式显示一个时间戳，示例如下。

```
mysql> SELECT FROM_UNIXTIME(UNIX_TIMESTAMP(), '%D %M %Y at %h:%i:%s');
+-----+
```

该查询的结果如下所示。

```
+-----+
| FROM_UNIXTIME('1326247953') |
+-----+
| 2012-01-10 21:12:33          |
+-----+
1 row in set (0.00 sec)
```

可以使用DATE\_FORMAT()函数的格式选项，以更好看的方式显示一个时间戳，示例如下。

```
SELECT FROM UNIXTIME(UNIX_TIMESTAMP(), '%D %M %Y at %h:%i:%s');
```

执行这条查询语句的时候，得到如下的结果。

```
+-----+
| FROM_UNIXTIME(UNIX_TIMESTAMP(), '%D %M %Y at %h:%i:%s') |
+-----+
| 10th January 2012 at 09:15:38 |
+-----+
1 row in set (0.00 sec)
```

## 16.12 小结

在本章中，我们学习了SQL的基本知识，从表的创建到操作记录。表创建命令需要3块重要的信息：表名、字段名和字段定义。字段定义很重要，因为一个设计良好的表有助于加快数据库的速度。MySQL有3个不同种类的数据类型：数字、日期和时间以及字符串。

INSERT命令用来向一个表添加记录，命令指定了要填充的表和列，并且随后定义了值。当把值放入到INSERT语句时，字符串必须用单引号或双引号括起来。SELECT命令用来从特定的表获取记录。\*字符使得我们能够容易地选择表中记录的所有字段，但是，我们也可以指定特定的列名。如果结果集太长，而我们指定了开始位置以及要返回的记录数目，那么LIMIT子句会提供一种简单的方法来提取需要的结果。要排序结果，可以使用ORDER BY子句来选择要排序的列。排序可以对整数、日期和字符串执行，按照升序或降序排序，默认的顺序是升序。如果没有指定顺序，结果会按照它们在表中的顺序来显示。

如果没有指定顺序，结果将会按照它们在表中出现的顺序来显示。我们可以使用WHERE子句来测试条件的有效性，从而挑选和选择要返回的记录。比较操作符或逻辑操作符可以在WHERE子句中使用，并且有时候两种类型都用以组成复合语句。在一条语句中从多个表选择记录也是比较高级的，这种类型的语句叫做JOIN，需要提前思考和规划以得到正确的结果。JOIN的常见类型是INNER JOIN、LEFT JOIN和RIGHT JOIN，尽管MySQL支持很多种不同类型的JOIN。我们还学习了，在操作多个表的时候可以使用子查询而不使用JOIN。

UPDATE和REPLACE命令用来修改MySQL表中已有的数据。

UPDATE用于改变特定列中的值以及根据特定的条件来改变多条记录中的值。REPLACE是INSERT语句的一个变体，它删除掉一条具有匹配的主键的记录然后重新插入新记录。在使用UPDATE来改变一个列中的值时要小心，因为忘了添加条件将会导致表中的所有记录的给定列都被更新。

DELETE语句很简单，它从表中删除记录，这也会有风险，因此，要确保只把DELETE权限授予那些可以担负责任的用户。可以在使用DELETE的时候指定条件，以便只有当WHERE子句中的一个特定表达式为true的时候才删除记录。另外，可以使用一条LIMIT子句来删除表中记录的一个较小的集合。如果我们有一个特别大的表，删除部分比删除大表中的每条记录的资源密集性要小。

我们介绍了在字符串、日期和时间上执行操作的MySQL函数。如果我们在MySQL有一个字符串要连接起来或者想要计算字符数，可以使用CONCAT()、CONCAT\_WS()和LENGTH()函数。要填充字符串或者删除字符串中的填充，使用RPAD()、LPAD()、TRIM()、LTRIM()和RRIM()来得到想要的字符串。还可以使用LOCATE()、SUBSTRING()、LEFT()和RIGHT()函数在一个字符串中查找另一个字符串的位置，或者返回一个给定字符串的一部分。像LCASE()、UCASE()、REPEAT()和REPLACE()这样的函数，也会返回最初的字符串的变体。MySQL内建的日期和时间函数，可以内部地格式化日期和时间以及执行日期和时间算术，这确实缓解了应用程序的负担。用于DATE\_FORMAT()的格式化选项，提供了一种简单的方法来从任何类型的日期字段产生一个自定义的显示字符串。DATE\_ADD()和DATE\_SUB()函数以及它们的众多可用

时间间隔类型，可以帮助我们确定过去或未来的日期和时间。

此外，像DAY()、WEEK()、MONTH()和YEAR()这样的函数，对于提取日期的部分以用于WHERE或ORDER BY子句也是很有用的。

## 16.13 Q&A

**Q:** 可以用什么字符串来命名表和字段？哪些字符是受到限制的？

**A:** 数据库、表和字段名的最大长度是64个字符。任何可以用于目录名或文件名的字符，也都可以用于数据库名和表名，除了/和.。这些限制是有必要的，因为MySQL在你的文件系统中创建了目录和文件，这些与数据库名和表名对应。除了长度，字段名方面没有字符限制。

**Q:** 我可以在一条语句中使用多个函数吗？例如，把一个连接的字符串变为全部大写。

**A:** 当然可以，只要别忘了开始括号和结束括号。下面这个例子展示了如何把master\_name表中的firstname和lastname连接起来并变成大写。

```
SELECT UCASE(CONCAT_WS(' ', firstname, lastname)) FROM master_name;
```

结果会如下所示。



```

+-----+
| UCASE(CONCAT_WS(' ', firstname, lastname)) |
+-----+
| JOHN SMITH |
| JANE SMITH |
| JIMBO JONES |
| ANDY SMITH |
| CHRIS JONES |
| ANNA BELL |
| JIMMY CARR |
| ALBERT SMITH |
| JOHN DOE |
+-----+
9 rows in set (0.00 sec)

```

如果只想把lastname变为大写，则使用如下语句。

```
SELECT CONCAT_WS(' ', firstname, UCASE(lastname)) FROM master_name;
```

结果会如下所示。

```

+-----+
| CONCAT_WS(' ', firstname, UCASE(lastname)) |
+-----+
| John SMITH |
| Jane SMITH |
| Jimbo JONES |
| Andy SMITH |
| Chris JONES |
| Anna BELL |
| Jimmy CARR |
| Albert SMITH |
| John DOE |
+-----+
9 rows in set (0.00 sec)

```

## 16.14 实践练习

实践练习是设计用来帮助你预料可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 练习题

1. 整数56678685可以是什么数据类型？
2. 如何定义一个只包含如下字符串的信息：apple、pear、banana和cherry？
3. 从一个表中选择前25条记录的LIMIT子句是什么？选择接下来的25条记录的LIMIT子句呢？
4. 如何使用LIKE来进行一个字符串的比较，匹配名字为“John”或“Joseph”？
5. 如何显式地引用名为table1的表中的一个名为id的字段？
6. 编写一条SQL语句，它连接orders和items\_ordered这两个表，每个表都有一个order\_id主键。从orders表中，选择如下的字段order\_name和order\_date。从items\_ordered表中，选择item\_description字段。
7. 编写一条SQL查询来查找一个子字符串“grape”在一个字符串“applepearbananagrape”中的开始位置。
8. 编写一条查询语句，它从字符串“applepearbananagrape”中选择最后的5个字符。

## 解答

1. MEDIUMINT, INT或BIGINT。

2. ENUM ('apple', 'pear', 'banana', 'cherry')

或

SET ('apple', 'pear', 'banana', 'cherry')。

3. LIMIT 0, 25和LIMIT 25, 25。

4. LIKE 'Jo%'。

5. 在查询中使用table1.id而不是id。

6. SELECT orders.order\_name, orders.order\_date,  
items\_ordered.item\_description FROM orders LEFT JOIN  
items\_ordered ON orders.order\_id = items\_ordered.id;

7. SELECT LOCATE('grape', 'applepearbananagrape');

8. SELECT RIGHT("applepearbananagrape", 5);

## 思考题

花点时间来创建某些示例表并且练习使用基本的INSERT和SELECT命令。

## 第17章 使用MySQL中的事务和存储过程

在本章中，你将学到：

- 事务的基础知识以及如何在MySQL中使用它们。
- 存储过程的基础知识以及如何在MySQL中创建和访问它们。

在前一章中，我们学习了SQL的基本知识以及如何使用MySQL命令行界面来执行查询和获取结果。仅仅这些知识，我们可以成功地完成在本书其他各章中建立的项目。然而，当我们要继续深入学习，并且考虑构建适合在企业环境中使用的应用程序的时候，可能需要更多高级的方法，以便维护数据的完整性并且增强应用程序和MySQL的通信。

尽管本书其他的章节不会包含本章中的元素，即本章保持示例尽可能地简单，以便你的基础知识足够牢固；但我们可以很容易地自行更新代码，来包含在本章所学习的信息。

## 17.1 什么是事务

数据库事务只是必须按照如下方式执行的一组查询：如果其中有一个查询没有执行完，那么所有的查询都将失败。例如，假设你有3个SQL语句为一组的查询，第二个查询取决于第一个查询的结果，而第三个查询取决于第二个查询的结果。如果第二个查询失败，你需要有一种方法来取消第一个查询的结果；类似的，如果第三个查询失败，你需要取消第一个查询和第二个查询的结果。

通过在数据库驱动的应用程序中建立事务性过程，我们可以确保持存储在数据库中的数据完整性。下面的各个小节介绍了通过命令行界面和PHP函数来使用事务的过程。

### 提示：

当使用InnoDB存储引擎的时候，MySQL中事务性的表才是可用的。从MySQL 5.5.5开始，InnoDB是表的默认存储引擎（参见<http://dev.mysql.com/doc/refman/5.5/en/innodb-default-se.html> 了解更多信息）。如果你的表是MyISAM类型的，它们将不具有事务性。

### 17.1.1 事务中使用的基本语法

当考虑在MySQL中使用事务的时候，我们还需要理解如下的关键技术。

- **COMMIT** ——这个命令出现在事务中的一系列查询的最后，只有在所有的查询成功地执行之后才执行此命令。
- **ROLLBACK** ——当事务中的系列查询的一个或多个失败时将使用

这个命令，并且把相关的表恢复到事务之前的状态。

回顾一下前面使用的例子，即3个彼此依赖的查询，MySQL命令行界面中的一系列事件如下所示。

1. 执行BEGIN命令开始一个新的事务。
2. 从table1中选择一个值以插入到table2中。
3. 如果没有从table1中选择一个值，执行一条ROLLBACK命令，以确保事务结束并且表返回到之前的状态。
4. 如果从table1中选择了值，将这个值插入到table2。
5. 如果向table2中插入一条记录失败，执行一条ROLLBACK命令，以确保事务结束并且表返回到之前的状态。
6. 如果一个值插入到table1中，再把这个值插入到table2。
7. 如果把一条记录插入到table3失败，执行一条ROLLBACK命令，以确保事务结束并且表返回到之前的状态。
8. 如果一条记录成功插入到table3，执行一条COMMIT命令，确保事务结束并且这个表正确地更新。

要了解有关MySQL中的事务的内部工作机制，请参考位于<http://dev.mysql.com/doc/refman/5.0/en/transactionalcommands.html> 的MySQL手册。



在下面一节中，我们将看到涉及到库存和销售记录的表使用事务的例子。

### 17.1.2 使用事务的例子

假设你已经创建了一个在线的商店，它拥有保存了库存、销售记录和销售记录的细目的数据库表。CREATE语句如下所示。

```
CREATE TABLE store_inventory (  
    id int not null primary key auto_increment,  
    item_name varchar(50),  
    item_price float(6,2),  
    item_qty int  
) ENGINE=InnoDB;  
  
CREATE TABLE store_orders (  
    id int not null primary key auto_increment,  
    purchaser_name varchar(50),  
    purchase_date datetime  
) ENGINE=InnoDB;  
  
CREATE TABLE store_orders_items (  
    id int not null primary key auto_increment,  
    order_id int,  
    inventory_id int,  
    item_qty int  
) ENGINE=InnoDB;
```

在这个例子的store\_inventory表中，我们可以找到如下的两条记录。

id	item_name	item_price	item_qty
1	Great Book	19.99	10
2	Awesome CD	9.99	20

如果一个顾客想要通过你的网上商店购买两个Great Books和一个Awesome CD，过程如下所示。

1. 用户完成一个在线表单并且试图为购买而进行支付，因此，执行一条BEGIN命令以使一个事务成为结账脚本的一部分，如下所示。

```
BEGIN;
```

2. 把store\_inventory表中的商品的数量减去相应的数目。

```
UPDATE store_inventory SET item_qty = item_qty - 2 WHERE id = 1;  
UPDATE store_inventory SET item_qty = item_qty - 1 WHERE id = 2;
```

3. 向store\_orders表添加一条记录：

```
INSERT INTO store_orders (purchaser_name, purchase_date)  
VALUES ('John Smith', now());
```

4. 如果添加记录失败，执行一条ROLLBACK命令来重置可供交易的商品的数量，如下所示。

```
ROLLBACK;
```

5. 如果添加记录成功，获取刚刚添加的记录的ID，并且通过在store\_orders\_lineitems表中插入记录，从而在向销售记录添加细目的查询中用到这个ID，如下所示。

```
INSERT INTO store_orders_items (order_id, inventory_id, item_qty)  
VALUES ('1', '1', '2');  
INSERT INTO store_orders_items (order_id, inventory_id, item_qty)  
VALUES ('1', '2', '1');
```

6. 如果添加记录失败，执行一条ROLLBACK命令来恢复商品可供交易的数量并且把store\_orders中的记录删除，如下所示。

```
ROLLBACK;
```

7. 如果添加记录成功，但是后续的信用卡支付或者其他支付方式失败，执行一条ROLLBACK命令来恢复商品可供交易的数量，把

store\_orders中的记录删除，并且把store\_orders\_lineitems中的记录删除，如下所示。

```
ROLLBACK;
```

8. 如果成功地添加了一条记录，并且后续的信用卡支付或其他支付方式也成功了，执行一条COMMIT命令来确保所有的改变都存储并且事务结束，如下所示。

```
COMMIT;
```

当然，一个在线商店不会通过命令行界面直接和MySQL交互，但是，会通过像PHP这样的脚本语言来交互。但是，如果你理解了事务背后的过程，将PHP加入其中很简单，只需要执行前面列出的查询和命令。任何其他的PHP和MySQL的交互也没有什么不同，我们会在第18章中学习到这些。

除了我们将在第18章中学习的内容，如果想要在自己的脚本中使用事务，请确保浏览一下PHP手册中的如下这些函数的定义。

- **mysqli\_autocommit()** —[http://www.php.net/mysqli\\_autocommit](http://www.php.net/mysqli_autocommit)
- **mysqli\_commit()** —[http://www.php.net/mysqli\\_commit](http://www.php.net/mysqli_commit)
- **mysqli\_rollback()** —[http://www.php.net/mysqli\\_rollback](http://www.php.net/mysqli_rollback)

还是要强调一下，一定要看一下MySQL手册以了解关于事务的更多信息，参见<http://dev.mysql.com/doc/refman/5.5/en/sql-syntax-transactions.html>，尤其是讨论不能回滚（管理性操作）的事务类型，知道其中大多数内容是有用的。

## 17.2 什么是存储过程

简单地说，存储过程是存储在数据库服务器上而不是Web服务器上的一个SQL编写的过程。你可能会认为，不会在Web服务器上存储任何过程，但是实际上，任何包含SQL查询的脚本都作为一个过程存储在Web服务器上。例如，应用程序中选择、删除、更新或插入数据到表中的每个查询（我们辛辛苦苦输入到脚本中的代码），将会作为存储过程存储到数据库中，并且在脚本中引用。

在代码中使用存储过程的支持者指出，性能和可维护性是这么做的关键原因。

- 更好的性能 —— 存储过程作为一个预编译的SQL存在于数据库中，因此，一个典型的两步过程（编译和执行）变成了单步过程（执行）。
- 易于维护 —— 在一个地方（数据库中）维护一条语句，比在多个地方（如Web服务器的各个脚本中）维护一条语句要少花很多时间。此外，把所有这些语句存储到数据库中，而不是存储到Web服务器文档根目录下的实际的文本文件中，这就多了一条保护措施，以防止有人访问你的Web服务器上的文件，这样一来，他们所拥有的只是调用存储过程的查询，而不是过程本身的逻辑。

一个有用的存储过程的例子，就是用来生成某种类型（可能是财务数据、销售库存或者其他的类型）的一个报表的SQL查询，想象一下那将是包含了很多过程的一个复杂查询。创建一个存储过程，而不是这种

类型的一个查询，将会得到存储过程的性能好处。如果我们有一个在整个应用程序中频繁使用的简单的查询，为它创建一个存储过程，将会得到存储过程的易维护性的好处。不管存储过程是简单的还是复杂的，创建和使用它都要遵守同样的基本的过程。

如下定义的表将用于这个存储过程的例子中。

```
CREATE TABLE testSP (  
    id int not null primary key auto_increment,  
    field_name varchar(25),  
    date_added datetime  
) ENGINE=InnoDB;
```

这张用于测试的表中的记录如下。

id	field_name	date_added
1	Value 1	2012-01-23 09:40:24
2	Value 2	2012-01-24 09:40:24
3	Value 3	2012-01-25 09:40:24
4	Value 4	2012-01-26 09:40:24
5	Value 5	2012-01-27 09:40:24
6	Value 6	2012-01-30 09:40:24
7	Value 7	2012-01-31 09:40:24
8	Value 8	2012-02-01 09:40:24
9	Value 9	2012-02-02 09:40:24
10	Value 10	2012-02-14 09:40:24

要将这个示例表转换为存储过程，接下来，必须确保MySQL知道我们将要在存储过程中使用的分隔符。这个例子中使用“//”作为分隔符，因此，必须执行如下的查询。

```
DELIMITER //
```

创建一个基本的存储过程的语法如下。

```
CREATE PROCEDURE procedure_name () query //
```

对于这个例子，这个存储过程只是从testSP表选取所有的数据，我们把这个存储过程命名为sp1，定义如下。

```
CREATE PROCEDURE sp1 () SELECT * FROM testSP WHERE date_added BETWEEN  
DATE_SUB(NOW(), INTERVAL 7 DAY) AND NOW() //
```

要调用这个存储过程，使用CALL命令，示例如下。

```
CALL sp1 () //
```

这个存储过程（SELECT查询）的结果会返回给你，如下所示。

id	field_name	date_added
2	Value 2	2012-01-24 09:40:24
3	Value 3	2012-01-25 09:40:24
4	Value 4	2012-01-26 09:40:24
5	Value 5	2012-01-27 09:40:24
6	Value 6	2012-01-30 09:40:24

在第18章中，我们将学习使用PHP执行这些SQL查询的过程。显然，这短短的几页对于使用存储过程来说甚至算不上隔靴搔痒，本节只是介绍概念。其他的内容推荐阅读MySQL 手册中关于存储过程的内容，位于<http://dev.mysql.com/doc/refman/5.5/en/stored-routines.html>。

## 17.3 小结

这个简短的一章简单地介绍了在使用InnoDB存储引擎的时候，MySQL中事务处理的概念以及存储过程的使用。除了简单地介绍了这两个概念，本章还使用了真实的例子来描述这两个高级话题。本章的全部目标是：介绍一些在你自己开发企业级应用的时候应该理解的一些概念和术语。在练习本书中的示例的时候，记住这些知识，并且尝试找到方法来改进以前的基础查询。

## 17.4 Q&A

**Q:** 既然MySQL支持事务，我必须随时使用事务吗？

**A:** 不是的，特别是如果你的应用程序和站点的动态方面是为了动态显示数据，而不是为了动态插入数据的话，不一定要使用事务。此外，如果数据的插入不一定是和财务或库存相关的业务，也不必使用事务。换句话说，如果你没有使用事务并且一次插入或更新查询失败的话，你要确保这个失败无伤大雅，不会给你带来经济损失，也不会导致你失去重要客户的数据。



## 17.5 实践练习

实践练习是设计用来帮助你预料可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 练习题

1. 判断对错：MyISAM是MySQL中默认的、完全事务性的存储引擎。
2. 如果包含3步操作的事务中的第2步失败了，应该执行哪条命令？
3. 使用存储过程的两个优点是什么？

## 解答

1. 错。InnoDB是默认的存储引擎并支持完全事务。
2. ROLLBACK
3. 性能更好和易于维护。

## 思考题

1. 想想可能需要事务的其他一些数据库交互类型，并且，像本章中一样，创建你自己的表集合。尝试在你可能需要调用和回滚命令的所有逻辑位置使用它们。
2. 创建一个存储过程（及其所需的表），它报告在两个给定日期之间所发生的一个特定货物的所有销售事务。

## 第18章 使用PHP和MySQL交互

在本章中，你将学到：

- 如何使用**PHP**连接到**MySQL**。
- 如何通过**PHP**脚本插入和查询数据。

既然我们已经学习了PHP的基础知识和使用MySQL的基础知识，这就为学习二者之间的交互做好了准备。把PHP看作是通向MySQL的一个管道，我们在上一章学习到的命令就是将要在本章发送到MySQL的命令，只是这次我们使用PHP发送它们。

## 18.1 MySQL函数和MySQLi函数

如果你使用过PHP的旧版本，或者使用的是PHP的当前版本和MySQL的旧版本，可能会熟悉mysql\_\*函数组。你可能还需要在互联网上找到一些使用mysql\_\*函数组的示例代码。

然而，从MySQL 4.1.3版开始（现在已经过去7年了），MySQL数据库系统包含了在PHP中强制使用新通信方法的功能，这些方法全部包含在mysqli\_\*函数组中。

本章中的所有代码，以及本书其他部分的代码，都使用mysqli\_\*函数组。要了解详细信息，请参阅位于<http://www.php.net/mysqli>的PHP手册相关章节。

## 18.2 使用PHP连接MySQL

要成功地使用PHP函数和MySQL交互，必须在Web服务器能够连接到的一个位置（不一定必须和Web服务器位于相同的机器）运行MySQL。还必须创建一个MySQL用户（带有一个密码），并且必须知道想要连接的数据库的名字。如果遵从第2章和第4章的说明，我们应该已经做好了这些准备。如果PHP和MySQL托管于一个Internet服务器提供商，请在处理之前，通过系统管理员确保你使用了正确的用户名、密码以及数据库名。

在本章的所有示例脚本中，示例数据库的名字是testDB，示例用户是joeuser，而示例密码是somepass。使用这些脚本的时候，请用你自己的信息替换它们。

### 提示：

本章中的所有代码以及其他章节的代码都进一步反映了过程式的mysqli\_\*函数组的使用。也可以以面向对象的方式来使用这些函数，要了解这方面的更多信息，请阅读位于<http://www.php.net/mysqli>的PHP手册。如果你是从一种面向对象编程语言或者有面向对象思想的编程语言转向PHP的，我建议你阅读一下PHP手册中的面向对象功能介绍，并且在适当的时候用它来替换，从概念上讲，这些过程都是相似的。

然而，如果你是编程新手，或者还没有接触面向对象编程思想，在你的日常工作中学习并使用过程式方式是没问题的。在本书中，我继续使用过程式编程。因为，对于程序员新手理解过程来说，这已经证明是最好的方法。

### 18.2.1 进行连接

连接到MySQL的基本语法如下。

```
$mysqli = mysqli_connect("hostname", "username", "password", "database");
```

`$mysqli`的值是函数的结果，并且随后用在与MySQL通信的函数中。

以一个实际的示例值，连接代码如下所示。

```
$mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
```

程序清单18.1是连接脚本的一个有效例子。它在第2行创建了一个新的连接，然后，测试是否发生了一个错误。如果发生了错误，第5行显示出一条错误消息，并且使用`mysqli_connect_error()`函数来显示消息。如果没有发生错误，第8行显示一条消息，其中包含了调用`mysqli_get_host_info()`函数得到的主机信息。

程序清单18.1 一个简单的连接脚本

---

```
1: <?php
2: $mysqli = new mysqli("localhost", "joeuser", "somepass", "testDB");
3:
4: if (mysqli_connect_errno()) {
5:     printf("Connect failed: %s\n", mysqli_connect_error());
6:     exit();
7: } else {
8:     printf("Host information: %s\n", mysqli_get_host_info($mysqli));
9: }
10: ?>
```

---

把这个脚本保存为`mysqlconnect.php`，并且将其放置到Web服务器的文档区域。使用Web浏览器访问这个脚本，如果连接成功的话，将会看到如下所示的结果。

```
Host information: localhost via TCP/IP
```



你可能还会看到如下内容。

```
Host information: Localhost via UNIX socket
```

如果连接失败，将会显示一条错误消息。第5行通过 `mysqli_connect_error()` 函数产生一个错误，例子如下所示。

```
Connect failed: Access denied for user 'joeuser'@'localhost' (using password: YES)
```

然而，如果连接成功，第8行将显示 `mysqli_get_host_info()` 的输出，就像上面的例子一样。

尽管脚本执行完毕的时候这个连接已经关闭了，但是，显式地关闭连接是个好主意。我们可以在程序清单18.2的第9行看到如何做到这一点。

程序清单18.2 修改后的简单连接脚本

---

```
1: <?php
2: $mysqli = new mysqli("localhost", "joeuser", "somepass", "testDB");
3:
4: if (mysqli_connect_errno()) {
5:     printf("Connect failed: %s\n", mysqli_connect_error());
6:     exit();
7: } else {
8:     printf("Host information: %s\n", mysqli_get_host_info($mysqli));
9:     mysqli_close($mysqli);
10: }
11: ?>
```

---

但是在第5行之后，我们没有使用 `mysql_close()` 函数，这是因为如果执行了第5行，当初就没有建立连接。对于使用PHP连接MySQL的介绍就告一段落了。下一节介绍查询执行函数，这比简单地打开一个连接什么也不做要有趣得多。

## 18.2.2 执行查询

知道如何编写SQL并且学习了本章前面的基本知识，那么使用PHP执行MySQL查询的任务就进行一半了。PHP中的函数mysql\_query()用来向MySQL发送SQL查询。

在脚本中，首先进行连接，然后执行一个查询。程序清单18.3中的脚本创建了一个名为testTable的示例表。

程序清单18.3 创建一个表的脚本

```
1: <?php
2: $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
3:
4: if (mysqli_connect_errno()) {
5:     printf("Connect failed: %s\n", mysqli_connect_error());
6:     exit();
7: } else {
8:     $sql = "CREATE TABLE testTable
9:           (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
10:            testField VARCHAR(75))";
11:     $res = mysqli_query($mysqli, $sql);
12:
13:     if ($res === TRUE) {
14:         echo "Table testTable successfully created.";
15:     } else {
16:         printf("Could not create table: %s\n", mysqli_error($mysqli));
17:     }
18:
19:     mysqli_close($mysqli);
20: }
21: ?>
```

### 提示：

通过一个脚本执行查询的时候，SQL语句末尾的分号不是必须的。

在第8行到第10行组成SQL语句的文本并赋给\$sql变量。这种做法是

随意的，你甚至不需要把SQL查询的内容放到一个单独的变量中，这个例子中这么做只是为了使这一过程的各个步骤更清楚。

`mysqli_query`函数返回一个值，`true`或`false`，从第13行开始的`if...else`语句检查这个值。如果`$res`的值为`true`，就会在屏幕上显示一条成功的消息。如果我们通过命令行界面访问MySQL，以验证`testTable`表的创建，将会看到`DESCRIBE test Table`命令的如下输出。

Field	Type	Null	Key	Default	Extra
id	int(11)		PRI	NULL	auto_increment
testField	varchar(75)	YES		NULL	

如果是这种情况，恭喜你，你已经使用PHP在MySQL数据库中成功地创建了一个表。

然而，如果`$res`的值不为`true`，将会显示一条错误的消息，这条消息由`mysqli_error()`函数产生。

### 18.2.3 获取错误消息

花点时间熟悉一下`mysqli_error()`函数，它将会成为你的朋友。当和PHP `die()`函数（在遇到该函数的时候直接退出脚本）一起使用的时候，如果出现错误，`mysqli_error()`函数将会返回一条有用的错误信息。

例如，既然我们已经创建了一个名为`testTable`的表，就可以再次执行该脚本而没有任何错误。尝试再次执行该脚本，应该会在Web浏览器中看到类似下面的结果：

```
Could not create table: Table 'testtable' already exists
```

多么令人兴奋！继续学习下一节，开始向表中插入数据，我们很快将通过**PHP**获取信息并格式化它。

## 18.3 使用MySQL数据

插入、更新、删除和获取数据都围绕着使用`mysqli_query()`函数来执行我们在第16章学习过的基本的SQL查询。对于INSERT、UPDATE和DELETE查询，在查询执行后不需要额外的脚本，因为我们不要显示任何结果（除非想要这么做）。使用SELECT查询的时候，有几个选项用来显示查询所获取的数据。让我们从基本的插入数据开始，这样就有了可供访问的内容。

### 18.3.1 避免SQL注入

在程序清单18.3的创建表脚本中，SQL查询中使用的数据是直接编码到脚本中的。然而，你可能要构建的是动态Web站点或者基于Web的应用程序，这时候，你很可能是根据一个表单的用户输入或者其他的过程，向表中插入数据或者从表中查询数据。如果你没有留意用户输入，并且在查询中使用用户输入之前没有进行安全性检查，那么，你可能会遭受SQL注入式攻击。

当个别怀有恶意的人借机在你的表单字段中输入整个或部分SQL查询的时候，就会发生SQL注入式攻击；我们假设执行这些查询的时候，安全性会受到破坏，数据有暴露的潜在危险。

一篇知名的XKCD网络漫画《Little Bobby Tables》，很好地说明了SQL注入式攻击的问题。各大讨论论坛和其他编程相关的帮助站点经常引用这段漫画，并且在针对表单输入和查询相关的问题给出解答的时候，都配上相应的说明，“Don't forget Little Bobby Tables!”。可以

在<http://xkcd.com/327/> 看到这段漫画。

看下面的示例，它试图从一个叫做users的表收集用户信息，其中name字段与表单中输入的一个值匹配；这很像是一个Web登录过程。

```
SELECT * FROM users
WHERE name = '". $_POST['username_from_form']. "';
```

假设username\_from\_form字段中的值如下所示。

```
' or '1'='1
```

这会产生如下的一个完整查询。

```
SELECT * FROM users
WHERE name = ' ' or '1'='1';
```

这个查询总是会产生一个有效的响应，因为11总是为真。

你可能已经明白了，但是如果还没有，PHP手册中关于SQL注入式攻击的页面上还有更多的示例，参见

<http://www.php.net/manual/en/security.database.sql-injection.php>。在整个本书中，代码示例都限制了SQL注入式攻击的可能性，只有一个例外，那就是显示错误消息。随着你不断学习，并且在一个开发环境而不是产品环境中操作，我建议你将在错误消息打印到屏幕上，以便理解发生了什么（或者没有发生什么）。在产品环境中，你应该隐藏错误消息，特别是当错误消息显示出数据库用户或表的名字的时候。

当你掌握了以本书所介绍的过程式方式使用MySQL和PHP的概念之后，看一下PDO(PHP Data Objects)抽象层，以进一步巩固你的产品应用程序，参见[http:// www. php. net/ manual /en/book.pdo.php](http://www.php.net/manual/en/book.pdo.php)。一个好的开始的地方，可能是关于预定义语句和存储过程的部分，参见<http://>

[www.php.net/manual/en/pdo.prepared-statements.php](http://www.php.net/manual/en/pdo.prepared-statements.php)。

### 18.3.2 使用**PHP**插入数据

在这个阶段，插入数据的最简单的方法就是直接编写INSERT语句，如程序清单 18.4所示。

程序清单18.4 插入一条记录的脚本

---

```
1: <?php
2: $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
3:
4: if (mysqli_connect_errno()) {
5:     printf("Connect failed: %s\n", mysqli_connect_error());
6:     exit();
7: } else {
8:     $sql = "INSERT INTO testTable (testField) VALUES ('some value')";
9:     $res = mysqli_query($mysqli, $sql);
10:
11:     if ($res === TRUE) {
12:         echo "A record has been inserted.";
13:     } else {
14:         printf("Could not insert record: %s\n", mysqli_error($mysqli));
15:     }
16:
17:     mysqli_close($mysqli);
18: }
19: ?>
```

---

和程序清单18.3相比，这个脚本的唯一变化就是用来插入记录，而程序清单18.3中的脚本是用来创建表，对二者而言SQL查询都是在第8行存储到\$*sql*变量中，而文本都是在第12行和第14行修改。连接代码和执行查询的结构都是相同的，实际上，连接MySQL的大多数过程式代码都将会遵从相同类型的代码模板。

把这个脚本命名为*mysqlinsert.php*，然后将其放入到Web服务器中。运行这个脚本将会使*testTable*表增加额外的一行。要输入比脚本中

更多的多条记录，可以编写一个更长的直接编码SQL语句列表并且多次使用mysql\_query()函数来执行这些语句，或者可以为记录添加脚本创建一个基于表单的界面。

要为这个脚本创建表单，确实只需要一个字段，因为id字段可以自动增加。表单的动作就是记录添加脚本的名字，我们把它叫做insert.php。HTML表单看上去如程序清单18.5所示。

程序清单18.5 一个插入表单

---

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>Record Insertion Form</title>
5: </head>
6: <body>
7: <form action="insert.php" method="POST">
8: <p><label for="testfield">Text to Add:</label><br/>
9: <input type="text" id="testfield" name="testfield" size="30" /></p>
10: <button type="submit" name="submit" value="insert">Insert Record</button>
11: </form>
12: </body>
13: </html>
```

---

把这个文件保存为insert\_form.html，并且将其放置到Web服务器的文档根目录下。接下来，创建程序清单18.6所示的insert.php脚本。表单中输入的值将会通过一个名为\$\_POST ['testfield']的变量来替换SQL查询中直接编码的值。

程序清单18.6 和表单一起使用的一个插入脚本



---

```
1: <?php
2: $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
3:
4: if (mysqli_connect_errno()) {
5:     printf("Connect failed: %s\n", mysqli_connect_error());
6:     exit();
7: } else {
8:     $clean_text = mysqli_real_escape_string($mysqli, $_POST['testfield']);
9:     $sql = "INSERT INTO testTable (testField)
10:         VALUES ('".$clean_text."')";
11:     $res = mysqli_query($mysqli, $sql);
12:
13:     if ($res === TRUE) {
14:         echo "A record has been inserted.";
15:     } else {
16:         printf("Could not insert record: %s\n", mysqli_error($mysqli));
17:     }
18:
19:     mysqli_close($mysqli);
20: }
21: ?>
```

---

这个脚本和程序清单18.4中的脚本之间的唯一差别在第8行，其中，表单的输入进行加强以避免SQL注入，并且在第10行，我们使用了安全的\$clean\_text字符串来替换前面示例中所使用的直接编码的文本字符串。为了使得输入更安全，我们使用了mysqli\_real\_escape\_string()函数，这个函数需要已经建立了连接，因此将它放到了if...else语句的else部分中。

把这个脚本保存为insert.php，并且将其放置在Web服务器的文档根目录下。在浏览器中访问所创建的这个HTML表单。它看上去如图18-1所示。

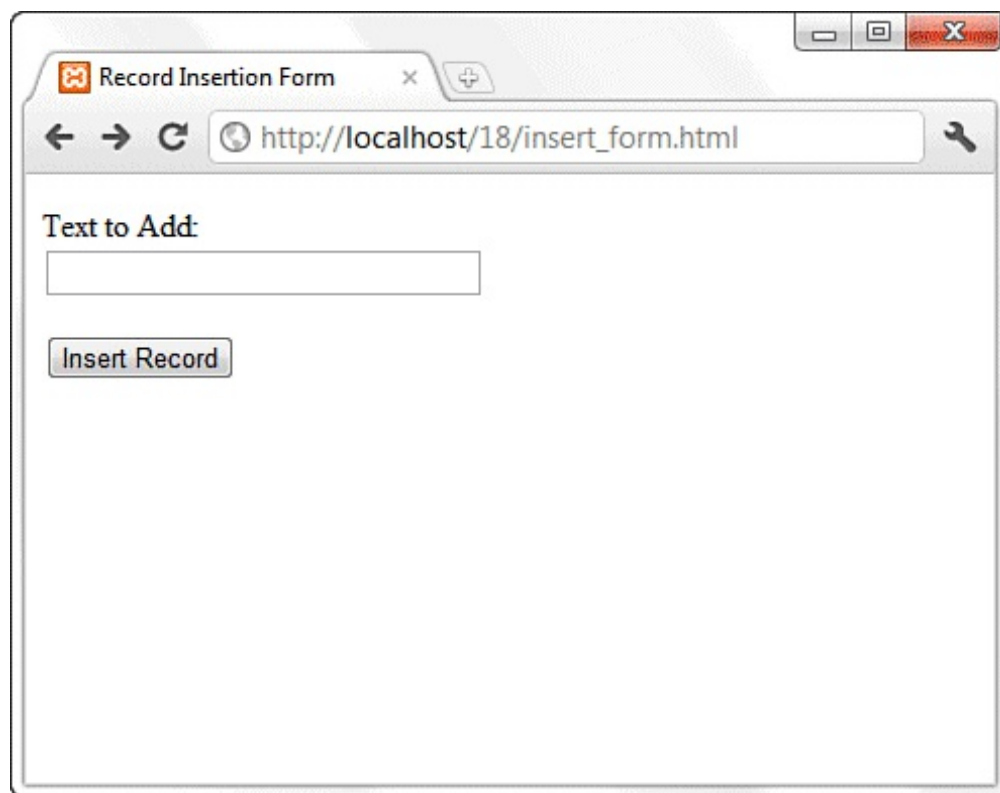


图18-1 用来添加一条记录的HTML表单

在Text to Add字段输入一个字符串，如图18-2所示。

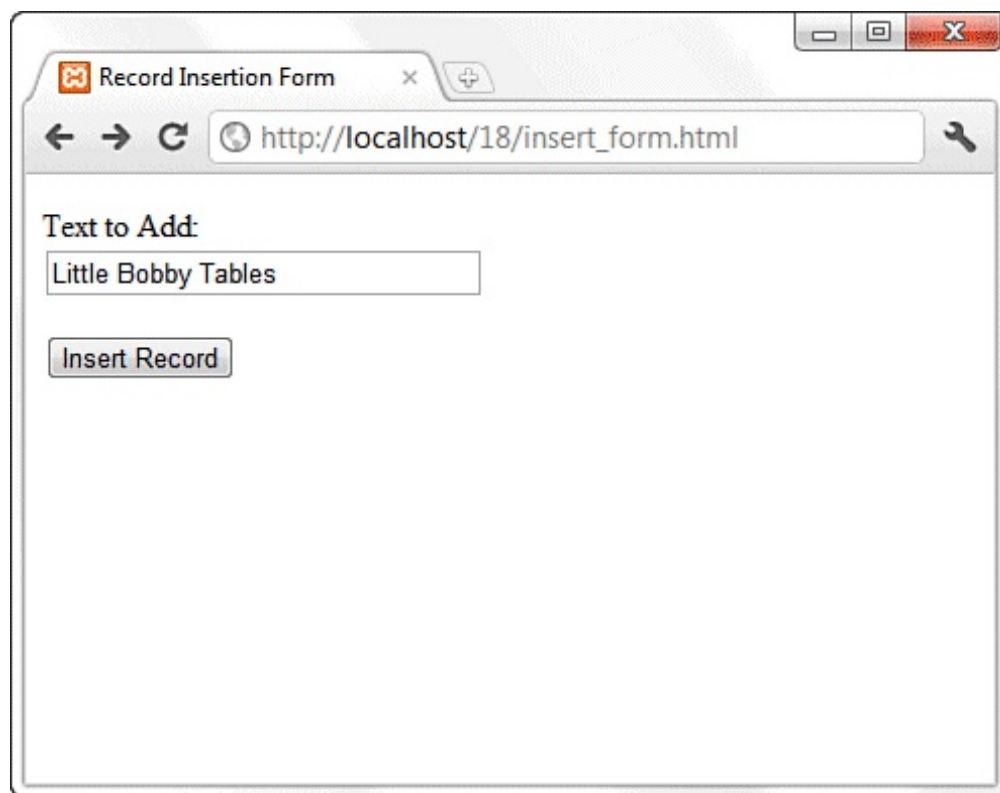


图18-2 表单字段中输入的文本

最后，点击Insert Record按钮来执行insert.php脚本，并且插入记录。如果成功，将会看到如图18-3所示的结果。

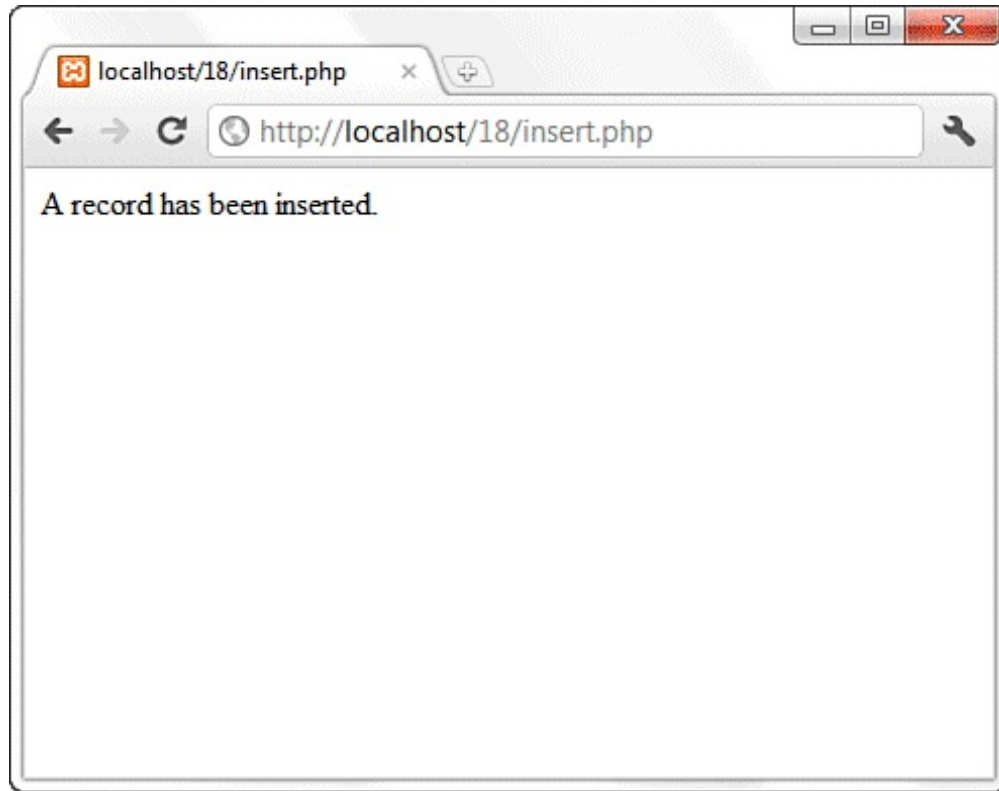


图18-3 记录已经成功添加

为了验证我们的工作，可以使用MySQL命令行界面来查看表中的记录。

```
SELECT * FROM testTable;
```

输出应该如下所示：

```
+-----+-----+
| id | testField |
+-----+-----+
| 1 | some value |
| 2 | Little Bobby Tables |
+-----+-----+
2 rows in set (0.00 sec)
```

接下来，我们将学习如何使用PHP获取和格式化结果，而不只是通过命令行。

### 18.3.3 使用PHP获取数据

由于已经在testTable表中有了一些数据行，我们可以编写一个PHP脚本来获取这些数据。从SQL基础知识开始，就介绍了编写一个脚本来执行一条SELECT查询语句。但是我们这里的SELECT语句不要产生太多的结果数据，我们只要得到行数，要做到这一点，就必须使用mysqli\_num\_rows()函数（参见程序清单18.7中的第12行）。

程序清单18.7 获取数据的脚本

---

```
1: <?php
2: $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
3:
4: if (mysqli_connect_errno()) {
5:     printf("Connect failed: %s\n", mysqli_connect_error());
6:     exit();
7: } else {
8:     $sql = "SELECT * FROM testTable";
9:     $res = mysqli_query($mysqli, $sql);
10:
11:     if ($res) {
12:         $number_of_rows = mysqli_num_rows($res);
13:         printf("Result set has %d rows.\n", $number_of_rows);
14:     } else {
15:         printf("Could not retrieve records: %s\n", mysqli_error($mysqli));
16:     }
17:
18:     mysqli_free_result($res);
19:     mysqli_close($mysqli);
20: }
21: ?>
```

---

把这个脚本保存为count.php，并放置到Web服务器的文档目录下，然后通过Web服务器访问它。将会看到如下的消息（实际上，这个数字将根据你插入到表中的记录的数目而有所不同）。

Result set has 4 rows.

第12行使用mysqli\_num\_rows()函数来获取结果集（\$res）中的行

数，然后，将这个值放入到一个名为\$number\_of\_rows的变量中。第13行在浏览器中显示这个数字，这个数字等于你在测试的时候所插入的记录数目。

**提示：**

这段程序中有一个前面程序没有用到的新函数，即第18行使用的mysql\_free\_result()函数。在使用mysql\_close()函数关闭连接之前使用mysql\_free\_result()函数，这确保了释放与查询和结果相关的所有内存以供其他脚本使用。

既然知道了表中有一些记录（根据输出来看，是4条记录），可以想象并获取这些记录的实际内容。我们可以用几种方法来做到这一点，但是，最简单的方法就是获取每一行构成一个数组。

我们将使用一条while语句来遍历结果集中的每条记录，把每个字段的值放入到一个特定的变量中，然后在屏幕上显示结果。

mysql\_fetch\_array()的语法如下。

```
$newArray = mysql_fetch_array($result_set);
```

下面使用程序清单18.8中的示例脚本。

程序清单18.8 获取数据并显示结果的一个脚本

---

```
1: <?php
2: $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
3:
4: if (mysqli_connect_errno()) {
5:     printf("Connect failed: %s\n", mysqli_connect_error());
6:     exit();
7: } else {
8:     $sql = "SELECT * FROM testTable";
9:     $res = mysqli_query($mysqli, $sql);
10:
11:     if ($res) {
12:         while ($newArray = mysqli_fetch_array($res, MYSQLI_ASSOC)) {
13:             $id = $newArray['id'];
14:             $testField = $newArray['testField'];
15:             echo "The ID is ".$id." and the text is: ".$testField."<br/>";
16:         }
17:     } else {
18:         printf("Could not retrieve records: %s\n", mysqli_error($mysqli));
19:     }
20:
21:     mysqli_free_result($res);
22:     mysqli_close($mysqli);
23: }
24: ?>
```

---

把这个脚本保存为select.php，并放置在Web服务器文档目录下，然后通过Web浏览器来访问它。你将会看到，对于输入到testTable中的每一条记录都将有一条消息，如图18-4所示。这些消息在第12行到第15行的While循环中创建。



图18-4 从MySQL选取记录

基本上，我们可以仅仅使用4个或5个MySQLi函数来创建一个完整的数据库驱动的应用程序。本章对于使用PHP和MySQL只是浅尝辄止，PHP中还有更多的MySQLi函数。

### 18.3.4 PHP中其他的MySQL函数

通过PHP中的MySQLi接口，有100多个可用的特定的MySQL函数。这些函数中的大多数只是改变了获取数据的方法，或改变了用来搜集所涉及的表结构的信息。在整本书中，特别是在稍后和项目相关的各章中，你将逐渐认识到PHP中更多的MySQL专用函数。然而，要获取函数的完整列表以及实际的例子，请访问位于<http://www.php.net/mysqli>的PHP手册的MySQLi部分。



## 18.4 小结

使用PHP和MySQL来创建动态的、数据库驱动的Web站点只是小菜一碟。别忘了，PHP函数基本上是通往数据库服务器的一个关口，用MySQL命令行界面输入的任何内容，都可以使用mysql\_query()函数输入。我们还学习了从一个表单接受用户输入的时候如何避免SQL注入。

要使用PHP连接到MySQL，你需要知道MySQL用户名、密码和数据库名。一旦连接上，可以使用mysql\_query()函数执行标准的SQL命令。如果已经执行了一条SELECT命令，可以使用mysql\_num\_rows()来计算结果集中返回的记录条数。如果想要显示得到的数据，可以在一个循环中使用mysql\_fetch\_array()来获取所有记录并将它们显示到屏幕上。

## 18.5 Q&A

**Q:** 能够在一个应用程序中同时使用**mysql\_\***和**mysqli\_\***函数吗？

**A:** 如果PHP安装后对两个库都支持，我们可以使用任何一组函数来和MySQL通信。然而，注意，如果对MySQL 4.1.3以后的版本使用**mysql\_\***函数组，可能不能够访问某些新的功能。此外，如果在整个应用程序中采用不一致的用法，维持和维护应用程序将会耗费更多时间并且效果也不理想。

## 18.6 实践练习

实践练习是设计用来帮助你预料可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 练习题

1. 用来进行PHP和MySQL之间的连接的主要函数是什么？需要哪些信息？
2. 哪个PHP函数获取一个MySQL错误消息的文本？
3. 哪个PHP函数用来计算一个结果集中的记录的条数？

## 解答

1. `mysqli_connect()`函数创建一个到MySQL的连接，需要主机名、用户名和密码。
2. `mysqli_error()`函数返回一条MySQL错误消息。
3. `mysqli_num_rows()`函数计算结果集中的记录数目。

## 思考题

1. 使用一个HTML表单和PHP脚本，创建一个表，其字段包含了一个人的姓氏和名字。创建另外一段脚本，来向表中添加记录。
2. 一旦表中有了记录，创建一段PHP脚本获取记录，并且按照姓氏的字母顺序显示这些记录。

## 第5部分 基本项目

第19章 管理一个简单的邮件列表

第20章 创建一个在线地址簿

第21章 创建一个简单的讨论论坛

第22章 创建一个在线商店

第23章 创建一个购物车机制

第24章 创建一个简单的日历

第25章 限制对应用程序的访问

第26章 记录并监视**Web**服务器活动

第27章 应用程序本地化

第28章 使用**XML**

## 第19章 管理一个简单的邮件列表

在本章中，你将学习到：

- 如何创建一个订阅/退订表单和脚本。
- 如何为发送消息创建一个前端。
- 如何创建发送消息的脚本。

本章提供了一个自己动手的小项目，后续章节还将有类似的项目，这些项目设计用来把我们的PHP和MySQL知识运用到一起。本章，我们将学习创建一个可管理的分发列表的方法，这个列表可以用来发送新闻组邮件或者其他任何东西，这些东西是要发送给数据库中的一个邮件地址列表的。

和本书中所有小的示例项目一样，这些项目可能和你想要用自己的PHP和MySQL知识构建的项目不完全相同。然而，我再怎么强调一点也不过分，这就是这个项目以及其他项目中展示的概念和示例，与你开发任何使用CRUD功能（创建、读取、更新和删除）的应用程序时所遇到概念和示例是类似的。

我们在本章中用到的邮件机制并不是用来替代邮件列表软件的，后者是专门设计来群发邮件的。我们在本章中构建的这种系统只能用于小型列表，即小于数百个邮件地址的列表。



## 19.1 开发订阅机制

我们在前面的各章中所学习的，都是创建任何产品所需的最重要的方面。在这个例子中，考虑我们的订阅机制所需要的如下元素。

- 一个保存Email地址的表。
- 用户添加或删除Email地址的方法。
- 发送消息的表单和脚本。

下面各节分别介绍每个项目元素。

### 19.1.1 创建subscribers表

在subscribers表中，我们真的只需要一个字段来存储用户的Email地址。然而，我们应该有一个ID字段，目的只是为了维护表之间的一致性，并且也因为在where子句中引用一个ID比引用一个长长的Email地址要简单得多。因此，在这个例子中，创建表的SQL语句将会如下所示。

```
CREATE TABLE subscribers (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    email VARCHAR (150) UNIQUE NOT NULL  
);
```

注意在Email的字段定义中UNIQUE的使用。这意味着，尽管id是主键，但是也不允许在Email字段出现重复。Email字段也是唯一键，但是id是主键。

通过命令行登录到MySQL并执行这个查询。在创建了表之后，执行一个DESC或DESCRIBE查询来验证表已经按照你的指定创建了，示

例如下。

```
mysql> DESC subscribers;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)       | NO   | PRI | NULL    | auto_increment |
| email | varchar(150)  | NO   | UNI | NULL    |                |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

既然在数据库中有表，我们就可以创建把值放入其中的表单和脚本了。

### 19.1.2 为共同函数创建一个包含文件

尽管这个过程中只有两个脚本，它们之中还是有一些共同函数，即数据库连接信息。要让脚本在这种情况下更为精简，把共同函数或代码段取出来，并放到一个文件中，然后通过我们在第13章学习的include语句将文件包含到其他脚本中。程序清单19.1包含了本章的脚本所共享的代码。

程序清单19.1 包含文件中的共同函数

---

```
1: <?php
2: // function to connect to database
3: function doDB() {
4:     global $mysqli;
5:
6:     //connect to server and select database
7:     $mysqli = mysqli_connect("localhost", "joeuser",
8:         "somepass", "testDB");
9:
10:    //if connection fails, stop script execution
11:    if (mysqli_connect_errno()) {
12:        printf("Connect failed: %s\n", mysqli_connect_error());
13:        exit();
14:    }
15: }
16: // function to check email address
17: function emailChecker($email) {
18:     global $mysqli, $safe_email, $check_res;
19:
20:     //check that email is not already in list
21:     $safe_email = mysqli_real_escape_string($mysqli, $email);
22:     $check_sql = "SELECT id FROM SUBSCRIBERS
23:         WHERE email = '".$safe_email."'";
24:     $check_res = mysqli_query($mysqli, $check_sql)
25:         or die(mysqli_error($mysqli));
26: }
27: ?>
```

---

第3到15行建立了一个函数doDB(), 这是一个简单的数据库连接函数。如果没有进行连接, 在调用这个函数的时候, 脚本将会退出; 否则, 它将让\$mysqli的值可供脚本其他部分使用。

第17行到第26行定义了一个名为emailChecker()的函数, 它接受一个输入并返回一个输出, 就像大多数函数所做的那样。当我们看到程序清单19.2的时候, 将在脚本中见到这个函数。

把这个文件保存为ch19\_include.php并将其放置到Web服务器上。在程序清单19.2中, 我们将会看到, 在需要的时候如何把这个文件包含到脚本中。

### 19.1.3 创建订阅表单

订阅表单实际上是一个名为manage.php的一体化表单和脚本，它会处理订阅和退订请求。程序清单19.2给出了manage.php的代码，它使用几个用户定义的函数来避免重复的代码，并且由此开始考虑创建自己的函数。代码看上去很长，但后面将逐行说明，并且这段代码的主要部分是HTML表单，因此别担心！

程序清单19.2 使用manage.php订阅和退订

---

```
1: <?php
2: include 'ch19_include.php';
3: //determine if they need to see the form or not
4: if (!$_POST) {
5:     //they need to see the form, so create form block
6:     $display_block = <<<END_OF_BLOCK
7:     <form method="POST" action="$_SERVER[PHP_SELF]">
8:
9:     <p><label for="email">Your E-Mail Address:</label><br/>
10:    <input type="email" id="email" name="email"
11:        size="40" maxlength="150" /></p>
12:
13:    <fieldset>
14:    <legend>Action:</legend><br/>
15:    <input type="radio" id="action_sub" name="action"
16:        value="sub" checked />
17:    <label for="action_sub">subscribe</label><br/>
18:    <input type="radio" id="action_unsub" name="action"
19:        value="unsub" />
20:    <label for="action_unsub">unsubscribe</label>
21:    </fieldset>
22:
23:    <button type="submit" name="submit" value="submit">Submit</button>
24:    </form>
25: END_OF_BLOCK;
26: } else if (($_POST) && ($_POST['action'] == "sub")) {
27:     //trying to subscribe; validate email address
28:     if ($_POST['email'] == "") {
29:         header("Location: manage.php");
30:         exit;
31:     } else {
32:         //connect to database
33:         doDB();
34:
35:         //check that email is in list
36:         emailChecker($_POST['email']);
37:
38:         //get number of results and do action
39:         if (mysqli_num_rows($check_res) < 1) {
40:             //free result
41:             mysqli_free_result($check_res);
42:
43:             //add record
44:             $add_sql = "INSERT INTO subscribers (email)
45:                 VALUES('".$_safe_email."')";
```

```

46:         $add_res = mysqli_query($mysqli, $add_sql)
47:         or die(mysqli_error($mysqli));
48:         $display_block = "<p>Thanks for signing up!</p>";
49:
50:         //close connection to MySQL
51:         mysqli_close($mysqli);
52:     } else {
53:         //print failure message
54:         $display_block = "<p>You're already subscribed!</p>";
55:     }
56: }
57: } else if (($_POST) && ($_POST['action'] == "unsub")) {
58:     //trying to unsubscribe; validate email address
59:     if ($_POST['email'] == "") {
60:         header("Location: manage.php");
61:         exit;
62:     } else {
63:         //connect to database
64:         doDB();
65:
66:         //check that email is in list
67:         emailChecker($_POST['email']);
68:
69:         //get number of results and do action
70:         if (mysqli_num_rows($check_res) < 1) {
71:             //free result
72:             mysqli_free_result($check_res);
73:
74:             //print failure message
75:             $display_block = "<p>Couldn't find your address!</p>";
76:             $display_block .= "<p>No action was taken.</p>";
77:         } else {
78:             //get value of ID from result
79:             while ($row = mysqli_fetch_array($check_res)) {
80:                 $id = $row['id'];
81:             }
82:
83:             //unsubscribe the address
84:             $del_sql = "DELETE FROM subscribers
85:                 WHERE id = ".$id;
86:             $del_res = mysqli_query($mysqli, $del_sql)
87:                 or die(mysqli_error($mysqli));
88:             $display_block = "<p>You're unsubscribed!</p>";
89:         }
90:         mysqli_close($mysqli);
91:     }
92: }
93: ?>
94: <!DOCTYPE html>
95: <html>
96: <head>
97: <title>Subscribe/Unsubscribe to a Mailing List</title>
98: </head>
99: <body>
100: <h1>Subscribe/Unsubscribe to a Mailing List</h1>
101: <?php echo "$display_block"; ?>
102: </body>
103: </html>

```

---

程序清单19.2可能有点长，但是它并不复杂。实际上，如果不是把用户定义的函数放在了ch19\_include.php中并在这段脚本的第2行包含它的话，程序清单19.2可能会更长。

第4行开始了脚本的主要逻辑。由于这个脚本执行了几个操作，我们需要首先确定它当前尝试的是哪一个操作。如果`$_POST`的结果为`false`，我们知道用户还没有提交表单，因此，我们必须为用户显示表单。

第6行到第25行，使用heredoc格式将一个字符串存储到`$display_block`变量中，创建了订阅/退订表单。在heredoc格式中，字符串分隔符可能是`<<<`后面的任何字符串标识符，只要最终的结束标识符单独一行就行了，正如你在这个示例的第25行中所见到的那样。

**提示：**

可以通过位于[http:// www.php.net/ manual/en/ language. types. string. php](http://www.php.net/manual/en/language.types.string.php) 的PHP手册，了解关于heredoc和其他字符串格式的更多信息。

在这个表单中，我们使用`$_SERVER[PHP_SELF]`作为动作（第7行），然后，创建了一个名为`email`的用于用户的邮件地址的文本字段（第9行到第11行），并且设置了一组单选按钮（第13行到第21行）来表示想要执行的任务。在字符串创建的最后，脚本跳出了`if...else`结构，跳到了第101行，并且显示变量`$display_block`中存储的HTML。这个表单显示如图19-1所示。

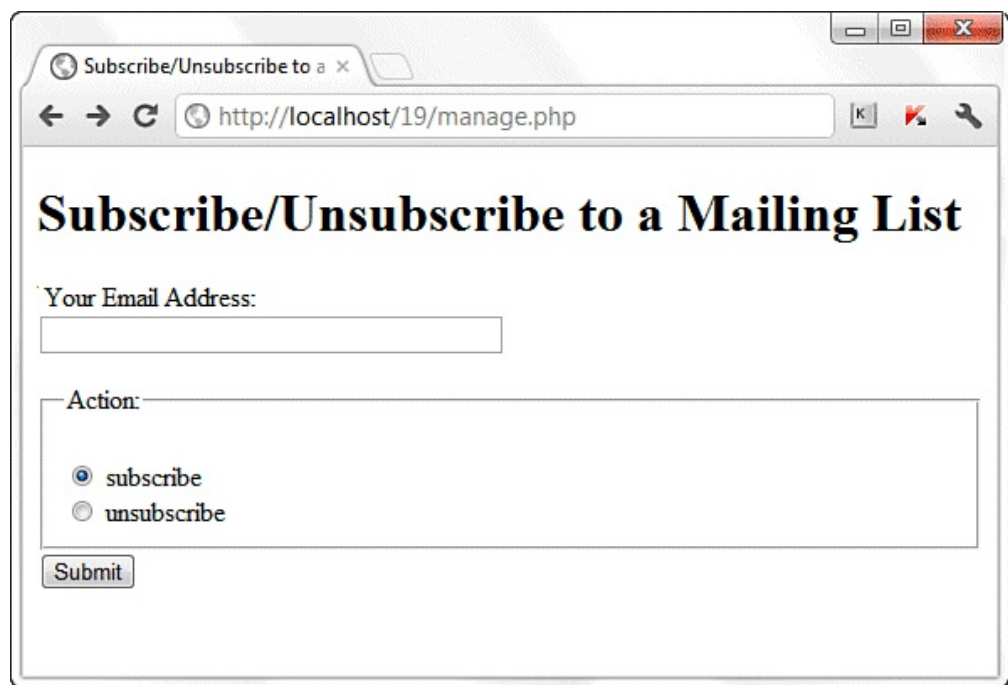


图19-1 订阅/退订表单

回到if...else结构，如果\$\_POST的结果为true，我们需要做一些事情。这有两种可能：对于表单中提供的Email地址的订阅或退订动作。我们需要看单选按钮组中的\$\_POST['action']的值来确定采取哪个操作。

在第26行，如果\$\_POST的结果为true并且\$\_POST['action']的值为“sub”，我们知道用户要订阅。为了订阅，用户需要一个Email地址，因此，在第28行到第30行检查这个地址。如果没有地址，用户将会重定向回到该表单。

然而，如果有地址，在第33行调用doDB()函数（存储在ch19\_include.php中）来连接数据库，以便可以发送查询。在第36行，我们调用第二个用户定义函数emailChecker()。这个函数接受一个输入（在本例中是\$\_POST['email']）并处理它。如果回头看看程序清单19.1的第21行到第25行，会看到emailChecker()函数中的代码执行了一个查



询，试图针对包含传递给函数的Email地址的记录，在subscribers表中找到一个id值。这个函数随后返回名为\$check\_res的结果集，以供在调用的脚本中使用。

**提示：**

注意，程序清单19.1中的两个用户定义函数的开始，都有全局变量的定义。这些变量由整个脚本共享，并且为此声明为全局的。

跳到下面的第39行，看看\$check\_res变量是如何使用的：把\$check\_res变量所引用的记录数目计算出来，以确定Email地址是否已经在表中存在。如果行数小于1，表明这个地址没有在列表中，并且它将会被添加到列表中。记录添加之后，响应在第44行到第48行存储。而失败消息（如果地址已经在表中存在）在第54行存储。此时，脚本跳出if...else结构，向下跳转到第101行，显示HTML。我们稍候将测试这一功能。

如果\$\_POST的结果为true并且\$\_POST['action']变量的值是“unsub”，就会出现输入的最后一种组合。在这种情况下，用户试图退订。为了退订，需要一个已有的Email地址，因此，我们在第59行到第61行检查它。如果没有给出地址，表单会回传给用户。

如果有地址，我们在第64行调用doDB()函数连接到数据库。然后，在第67行调用emailChecker()函数，它将再次返回结果集\$check\_res。在第70行，计算结果集中的记录数，从而来确定Email地址是否已经存在于表中。如果行数值小于1，地址没有在列表中，这样将无法退订。

在这个例子中，响应消息在第75行到第76行存储。然而，如果行数不小于1，用户将被退订（记录删除）并且在第84行到第88行存储响应。此刻，脚本跳出if...else结构，跳到第101行，输出HTML。

图19-2到图19-5给出了这个脚本根据选择的操作和数据库中的Email地址的状态所产生的各种结果。



图19-2 成功订阅



图19-3 订阅失败



图19-4 成功退订



图19-5 退订不成功

接下来，我们将创建给每个订阅者发送Email的表单和脚本。

## 19.2 开发邮件发送机制

准备好了订阅机制，我们可以在一个脚本创建一个基本的表单，这个脚本从表单提取内容，并且将其发送给subscribers表中的每个地址。下面是另一个完整的脚本，名为sendmymail.php，程序清单19.3给出其内容。

### 提示：

在尝试使用本节中的脚本之前，确保已经阅读了第11章中关于在php.ini文件中进行配置的部分，这是发送邮件所必需的。

程序清单19.3 向订阅者列表发送邮件

```
1: <?php
2: include 'ch19_include.php';
3: if (!$_POST) {
4:     //haven't seen the form, so display it
5:     $display_block = <<<END_OF_BLOCK
6:     <form method="POST" action="$_SERVER[PHP_SELF]">
7:
8:     <p><label for="subject">Subject:</label><br/>
9:     <input type="text" id="subject" name="subject" size="40" /></p>
10:
11:     <p><label for="message">Mail Body:</label><br/>
12:     <textarea id="message" name="message" cols="50"
13:         rows="10"></textarea></p>
13:     <button type="submit" name="submit" value="submit">Submit</button>
14:     </form>
15: END_OF_BLOCK;
16: } else if ($_POST) {
```

```

17:    //want to send form, so check for required fields
18:    if (($_POST['subject'] == "") || ($_POST['message'] == "")) {
19:        header("Location: sendmymail.php");
20:        exit;
21:    }
22:
23:    //connect to database
24:    doDB();
25:
26:    if (mysqli_connect_errno()) {
27:        //if connection fails, stop script execution
28:        printf("Connect failed: %s\n", mysqli_connect_error());
29:        exit();
30:    } else {
31:        //otherwise, get emails from subscribers list
32:        $sql = "SELECT email FROM subscribers";
33:        $result = mysqli_query($mysqli, $sql)
34:            or die(mysqli_error($mysqli));
35:
36:        //create a From: mailheader
37:        $mailheaders = "From: Your Mailing List
38:            <you@yourdomain.com>";
39:        //loop through results and send mail
40:        while ($row = mysqli_fetch_array($result)) {
41:            set_time_limit(0);
42:            $email = $row['email'];
43:            mail("$email", stripslashes($_POST['subject']),
44:                stripslashes($_POST['message']), $mailheaders);
45:            $display_block .= "newsletter sent to: ".$email."<br/>";
46:        }
47:        mysqli_free_result($result);
48:        mysqli_close($mysqli);
49:    }
50: }
51: ?>
52: <!DOCTYPE html>
53: <html>
54: <head>
55: <title>Sending a Newsletter</title>
56: </head>
57: <body>
58: <h1>Send a Newsletter</h1>
59: <?php echo $display_block; ?>
60: </body>
61: </html>

```

---

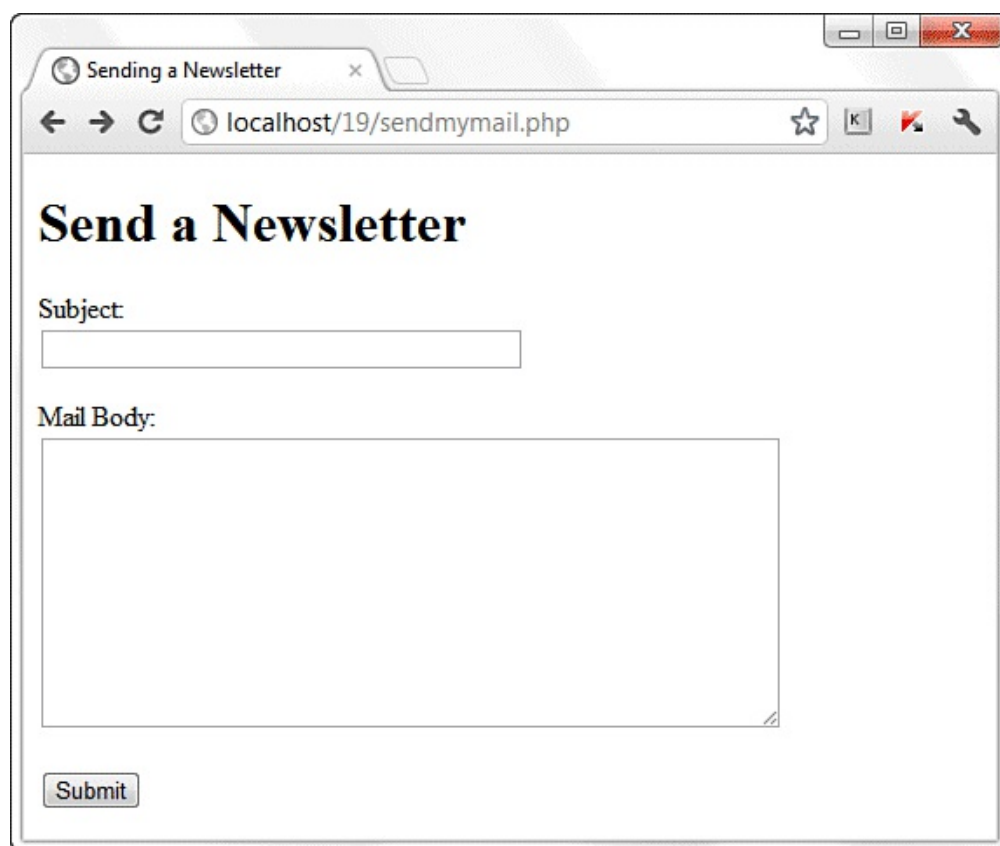
在程序清单19.3中，用户定义的函数的文件在第2行被包含进来。尽管在这个文件中只有数据库连接函数会用到，但把其他的函数包含在

这个文件中也无伤大雅。

脚本的主要逻辑从第3行开始，在那里，我们确定用户是否已经看到表单。如果\$\_POST变量为false，我们知道用户还没有提交表单，因此，我们必须显示页面。

第5行到第15行创建了向订阅者列表发送新闻组邮件的表单，它使用\$\_SERVER [PHP\_SELF]作为动作（第6行），创建一个叫做subject的文本字段来表示邮件的主题，并且创建了一个名为message的文本域来表示要发送的信件的正文。

在这里，脚本跳出了if...else构造，并且HTML显示出来。所显示的表单如图19-6所示。



The screenshot shows a web browser window with the title 'Sending a Newsletter'. The address bar displays 'localhost/19/sendmyemail.php'. The main content area features a form titled 'Send a Newsletter'. The form includes a 'Subject:' label followed by a single-line text input field, and a 'Mail Body:' label followed by a large multi-line text area. At the bottom left of the form is a 'Submit' button.

图19-6 发送大批邮件的表单

如果`$_POST`的结果不是`false`，脚本将会把表单发送到`subscribers`表中的邮件地址。在我们发送消息前，我们必须在第18行到第20行检查来自表单的两个必需项目：`$_POST['subject']`和`$_POST['message']`。如果这两个项目中有一个不存在，用户将会再次重定向表单。

如果所需项目具备了，脚本移动到第24行调用数据库连接函数。第33行执行一个查询，从`subscribers`表抓取所有的Email地址。尽管这些结果没有顺序，但是如果由于某种原因我们想要按照字母顺序送出邮件，可以在这里使用一条`order by`子句。

第37行到第38行创建了一个`From:`邮件标头，当邮件发送的时候，这个标头将用在未来的`while`循环中。这个标头确保了邮件看上去像是来自某一个人而不是一台机器，因为我们已经明确地在这个字符串中提供了一个值。开始于第40行的`while`循环，每次从结果集中提取一个邮件地址。在第41行，我们使用`set_time_limit()`函数把时间限制设置为0或者“no limit.”。这么做允许脚本运行所需要的那么长时间。

提示：

由于程序清单19.3中的所有脚本多次执行了`mail()`函数，它确实没有考虑到实际邮件列表软件中的排队因素，这是为减轻发送邮件服务器负担而设计的。使用`set_time_limit()`并不会缓解这一负担，它只是在时间可能超时之前允许脚本继续运行。

在第43行到第44行，使用`mail()`函数发送邮件，插入了来自相应的表单的值。第45行向屏幕显示了一条消息，以便告知谁应该接受邮件。在图19-7和图19-8中，我们看到了脚本的输出。





图19-7 邮件已经发送

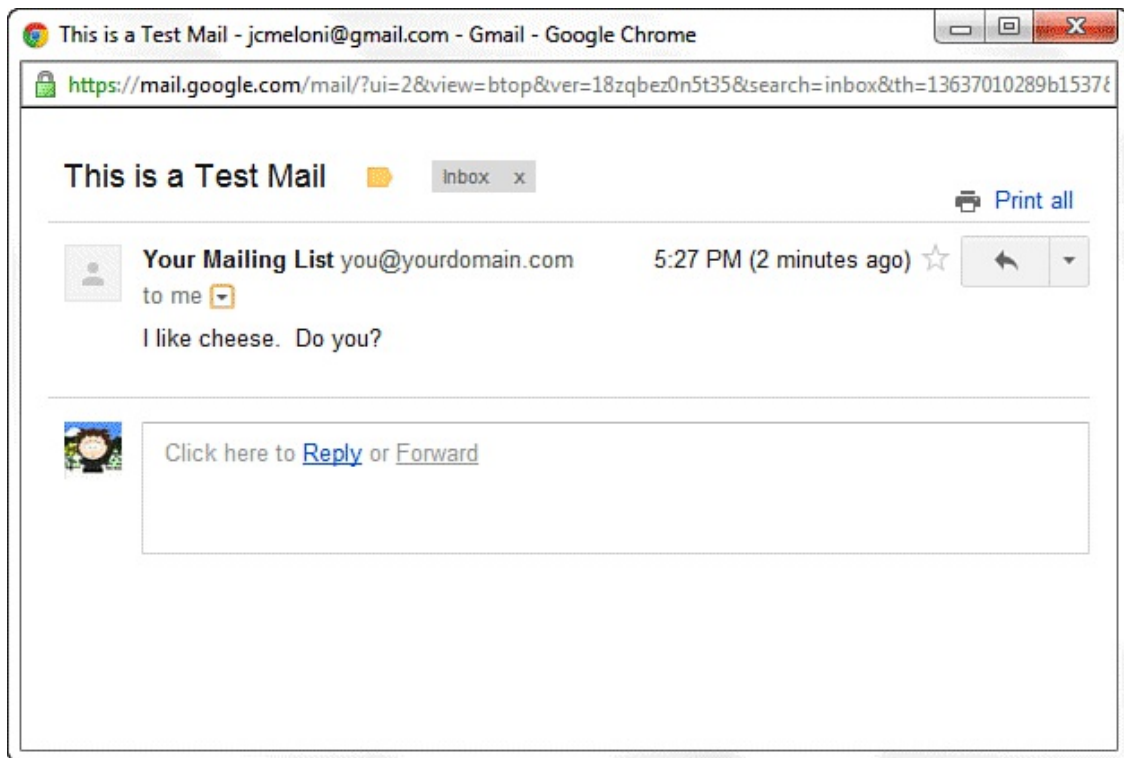


图19-8 邮件安全到达



## 19.3 小结

在这个动手实验的一章中，我们应用了基本的PHP和MySQL知识来创建一个个人邮件列表。包括数据库表的创建、订阅/退订机制，以及发送邮件的表单和脚本。

## 19.4 Q&A

**Q:** 如何缓解邮件服务器的负担？

**A:** 除了深入理解打包了的邮件列表软件，我们可以通过一个socket连接绕过mail()函数并直接和SMTP服务器对话。这样的例子在PHP手册的fsockopen()函数中可以找到(<http://www.php.net/fsockopen>), 在其他开发资源站点上也可以找到。

**Q:** 响应消息走到哪里？

**A:** 和任何Email一样（不只是按照本章中描述的方式发送的那些），响应消息去往我们在From:或Reply-to邮件标头中指定的任何地址。

## 19.5 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 什么函数发送邮件？
2. 为什么在程序清单19.1中`$mysqli`指定为一个全局变量。
3. 什么函数调用导致脚本执行它所需的那么长时间？

## 解答

1. 这不是一个很难的问题，`mail()`函数可实现。
2. 因为变量`$mysqli`在一个函数中创建并赋值，而这个函数被包含在供其他函数使用的一段脚本中，因此，该变量必须声明为全局的，以确保在创建它的函数之外也能使用它。
3. `set_time_limit(0)`。

## 思考题

1. 修改manage.php脚本，将用户的Email作为任何操作的响应消息的一部分而显示。
2. 修改sendmymail.php脚本，以添加额外的表单字段，它们将对消息字符串自身中的小节标题做出响应。记住，当表单提交的时候，这些字符串必须串成一个消息字符串，并将该字符串发送给mail()函数。

## 第20章 创建一个在线地址簿

在本章中，你将学到：

- 如何为一个在线地址簿创建关系表。
- 如何创建表单和脚本来添加和删除地址簿中的记录。
- 如何创建表单和脚本来浏览记录。

在这个动手实验的章节里，项目将创建一个可管理的在线地址簿。我们将学习创建相关的数据库表的方法，以及创建表单和脚本来添加、删除和浏览数据库记录的方法。

这些基本概念，是开发任何使用CRUD功能（创建、读取、更新和删除）的应用程序的基础，这一点和很多Web应用程序都是一样的。

# 20.1 规划和创建数据库表

当我们考虑一个地址簿的时候，有些明显的字段会涌入脑海：名字、地址、电话号码、邮件地址等等。然而，如果我们查看纸质的地址簿，可能会注意到一个人有多个条目。可能这个人有3个电话号码或者两个Email地址等等，或者任何与原始模板不相符的地方。在我们的在线地址簿中，一组相关的表将协助缓解信息的冗余和重复，并且允许我们以一致的视图来显示所有信息。

表20-1给出了用于在线地址簿的示例表及其字段名，我们将使用真正的SQL语句来创建这些表，但首先应该看看这些信息并尝试看看现有的关系。自己思考一下哪个字段应该是主键或唯一键。

表20-1 地址簿表和字段名

表 名	字 段 名
master_name	id, date_added, date_modified, f_name, l_name
address	id, master_id, date_added, date_modified, address, city, state, zipcode, type
Telephone	id, master_id, date_added, date_modified, tel_number, type
fax	id, master_id, date_added, date_modified, fax_number, type



email	id, master_id, date_added, date_modified, email, type
personal_notes	id, master_id, date_added, date_modified, note

注意和日期相关的字段的用法，每个表中都有一个date\_added和date\_Modified字段。这个字段将帮助你维护数据，我们可能在某个时刻要执行一个查询，把多少个月之前或多少年之前的所有记录删除掉，或者把在某段时间内没有更新过的所有记录删除掉。

正如将在如下的SQL语句中见到的，除了ID以及和日期相关的字段，master\_name表还有两个字段，这就是f\_name和l\_name，分别表示名和姓，id字段是主键。不需要其他的字段作为主键或唯一键，除非我们真的需要限制地址簿只能有一个John Smith、一个Mary Jones等。

#### 提示：

下面的语句中的文本字段的字段长度是任意的，你可以让它们尽可能地短，只要在字段类型允许的定义内。

下面的SQL语句创建了master\_name表。

```
CREATE TABLE master_name (
  id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  date_added DATETIME,
  date_modified DATETIME,
  f_name VARCHAR (75),
  l_name VARCHAR (75)
);
```

接下来，我们将创建辅助表，这些表都和master\_name表有关系。例如，address表具有自己的主键id字段以及date\_added和date\_modified字

段，以及用来建立关系的master\_id字段。

master\_id将等于master\_name表中的id字段，匹配到这是谁的地址。master\_id字段不是一个唯一键，因为一个人拥有多个地址条目绝对是合理的。我们在type字段中看到了这一点，这个字段定义为包含3个选项的一个枚举类型，这3个选项是home、work或other。一个人可以有三个类型中的一个或多个，因此，除了主键id，这个表中没有其他的键。假设这个特定的地址簿只包括美国的地址，我们用address、city、state和zipcode字段来简化这个表。

```
CREATE TABLE address (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    master_id INT NOT NULL,  
    date_added DATETIME,  
    date_modified DATETIME,  
    address VARCHAR (255),  
    city VARCHAR (30),  
    state CHAR (2),  
    zipcode VARCHAR (10),  
    type ENUM ('home', 'work', 'other')  
);
```

telephone、fax和email表也都是同一主题的变形，建表的SQL语句如下。

```

CREATE TABLE telephone (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    master_id INT NOT NULL,
    date_added DATETIME,
    date_modified DATETIME,
    tel_number VARCHAR (25),
    type ENUM ('home', 'work', 'other')
);
CREATE TABLE fax (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    master_id INT NOT NULL,
    date_added DATETIME,
    date_modified DATETIME,
    fax_number VARCHAR (25),
    type ENUM ('home', 'work', 'other')
);
CREATE TABLE email (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    master_id INT NOT NULL,
    date_added DATETIME,
    date_modified DATETIME,
    email VARCHAR (150),
    type ENUM ('home', 'work', 'other')
);

```

personal\_notes表也遵从同样的模式，只不过master\_id是一个唯一键并且只允许每人一条notes记录，SQL语句如下：

```

CREATE TABLE personal_notes (
    id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
    master_id INT NOT NULL UNIQUE,
    date_added DATETIME,
    date_modified DATETIME,
    note TEXT
);

```

既然已经创建了表，我们可以编写用来管理和浏览记录的表单和脚本了。

## 20.2 为共同函数创建一个包含文件

在第19章中，我们使用共同函数的一个包含文件使得脚本更为精简。在本章中也是这样。尽管共同函数文件中唯一的内容是数据库连接函数，这个过程还是能起到两个作用：让脚本更为精简，并且当数据库连接的信息发生变化的时候，不用在多个文件中修改这一信息。程序清单20.1包含了本章中的脚本共享的代码。

程序清单20.1 包含文件中的共同函数

---

```
1:  <?php
2:  function doDB() {
3:      global $mysqli;
4:
5:      //connect to server and select database; you may need it
6:      $mysqli = mysqli_connect("localhost", "joeuser",
7:          "somepass", "testDB");
8:
9:      //if connection fails, stop script execution
10:     if (mysqli_connect_errno()) {
11:         printf("Connect failed: %s\n", mysqli_connect_error());
12:         exit();
13:     }
14: }
15: ?>
```

---

第2行到第14行建立了第一个函数doDB()，这是一个简单的数据库连接函数。如果没有成功建立连接，在调用这个函数的时候，脚本将会退出；否则，它将让\$mysqli的值可供脚本其他部分使用。

把这个文件保存为ch20\_include.php并将其放置到Web服务器上。本章中的其他代码将会在脚本的前几行中包含这个文件。

## 20.3 创建一个菜单

在线地址簿将包含数种操作，因此，为链接创建一个菜单是有意义的。程序清单 20.2 为我们将在本章编写的所有脚本创建了一个简单的菜单，叫做mymenu.html。

程序清单20.2 地址簿菜单

---

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>My Address Book</title>
5: </head>
6: <body>
7: <h1>My Address Book</h1>
8:
9: <p><strong>Management</strong></p>
10: <ul>
11: <li><a href="addentry.php">Add an Entry</a></li>
12: <li><a href="delentry.php">Delete an Entry</a></li>
13: </ul>
14:
15: <p><strong>Viewing</strong></p>
16: <ul>
17: <li><a href="selentry.php">Select a Record</a></li>
18: </ul>
19: </body>
20: </html>
```

---

图20-1显示了程序清单20.2的输出。我们将按照顺序处理这些项中的每一个，从下一节的“Add an Entry”开始。

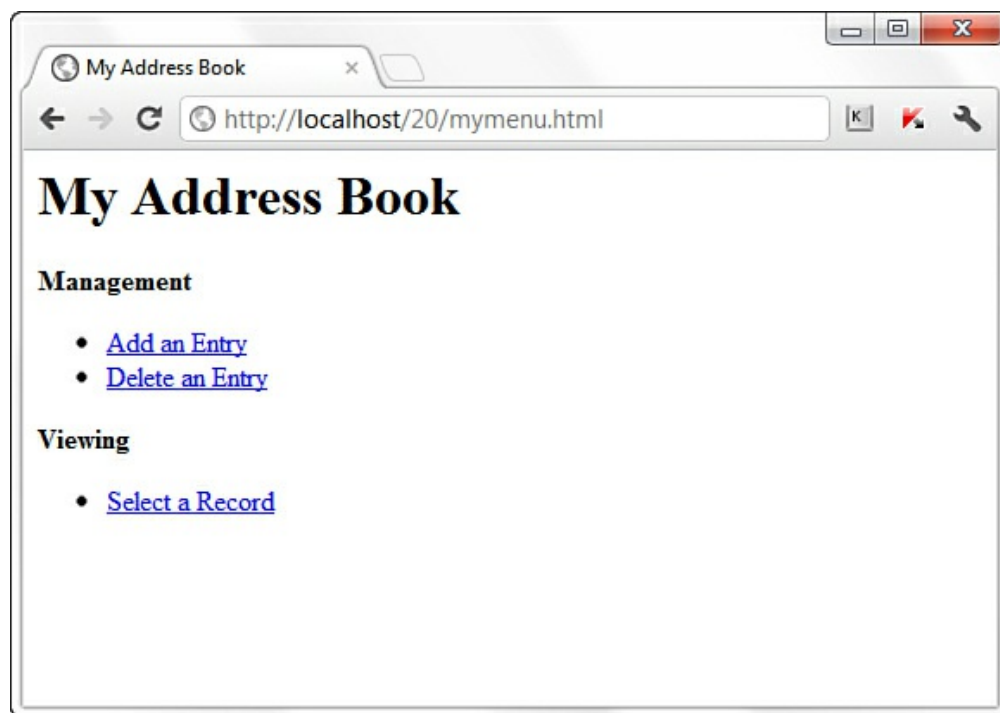


图20-1 地址簿菜单

## 20.4 创建记录添加机制

只是因为我们要潜在地添加信息到6个不同的表中，并不意味着表单或脚本会很庞大。实际上，脚本看上去和在前面的章节所创建的脚本没有太大差异，并且稍加练习，我们就能够让这些冗长的脚本更加流畅和高效。

在程序清单20.3中，你可以看到一个名为addentry.php的基本记录添加脚本，它有两部分：如果要显示表单应该做些什么（第4行到第89行）以及如果要提交表单应该采取什么动作（第91行到第187行）。第3行到第89行只是简单地把HTML表单的内容放入到一个名为\$display\_block的字符串中。

程序清单20.3 名为addentry.php的基本记录添加脚本

---

```
1:  <?php
2:  include 'ch20_include.php';
3:
4:  if (!$_POST) {
5:      //haven't seen the form, so show it
6:      $display_block = <<<END_OF_TEXT
7:      <form method="post" action="$_SERVER[PHP_SELF]">
8:      <fieldset>
9:      <legend>First/Last Names:</legend><br/>
10:     <input type="text" name="f_name" size="30"
11:         maxlength="75" required="required" />
12:     <input type="text" name="l_name" size="30"
13:         maxlength="75" required="required" />
14:     </fieldset>
15:
16:     <p><label for="address">Street Address:</label><br/>
17:     <input type="text" id="address" name="address"
18:         size="30" /></p>
19:
20:     <fieldset>
21:     <legend>City/State/Zip:</legend><br/>
22:     <input type="text" name="city" size="30" maxlength="50" />
23:     <input type="text" name="state" size="5" maxlength="2" />
24:     <input type="text" name="zipcode" size="10" maxlength="10" />
25:     </fieldset>
26:
27:     <fieldset>
28:     <legend>Address Type:</legend><br/>
29:     <input type="radio" id="add_type_h" name="add_type"
30:         value="home" checked />
31:         <label for="add_type_h">home</label>
32:     <input type="radio" id="add_type_w" name="add_type"
33:         value="work" />
34:         <label for="add_type_w">work</label>
35:     <input type="radio" id="add_type_o" name="add_type"
36:         value="other" />
37:         <label for="add_type_o">other</label>
38:     </fieldset>
```



```

39:
40:     <fieldset>
41:     <legend>Telephone Number:</legend><br/>
42:     <input type="text" name="tel_number" size="30" maxlength="25" />
43:     <input type="radio" id="tel_type_h" name="tel_type"
44:         value="home" checked />
45:         <label for="tel_type_h">home</label>
46:     <input type="radio" id="tel_type_w" name="tel_type"
47:         value="work" />
48:         <label for="tel_type_w">work</label>
49:     <input type="radio" id="tel_type_o" name="tel_type"
50:         value="other" />
51:         <label for="tel_type_o">other</label>
52:     </fieldset>
53:
54:     <fieldset>
55:     <legend>Fax Number:</legend><br/>
56:     <input type="text" name="fax_number" size="30" maxlength="25" />
57:     <input type="radio" id="fax_type_h" name="fax_type"
58:         value="home" checked />
59:         <label for="fax_type_h">home</label>
60:     <input type="radio" id="fax_type_w" name="fax_type"
61:         value="work" />
62:         <label for="fax_type_w">work</label>
63:     <input type="radio" id="fax_type_o" name="fax_type"
64:         value="other" />
65:         <label for="fax_type_o">other</label>
66:     </fieldset>
67:
68:     <fieldset>
69:     <legend>Email Address:</legend><br/>
70:     <input type="email" name="email" size="30" maxlength="150" />
71:     <input type="radio" id="email_type_h" name="email_type"
72:         value="home" checked />
73:         <label for="email_type_h">home</label>
74:     <input type="radio" id="email_type_w" name="email_type"
75:         value="work" />
76:         <label for="email_type_w">work</label>
77:     <input type="radio" id="email_type_o" name="email_type"
78:         value="other" />
79:         <label for="email_type_o">other</label>
80:     </fieldset>
81:
82:     <p><label for="note">Personal Note:</label><br/>
83:     <textarea id="note" name="note" cols="35"
84:         rows="3"></textarea></p>
85:
86:     <button type="submit" name="submit"
87:         value="send">Add Entry</button>
88:     </form>
89:     END_OF_TEXT;

```

在这里，先稍等片刻，确保你知道在上述程序清单中发生了什么。在浏览其他代码之前，注意用户定义函数的文件在第2行被包含了进来。正如已经提到的，这个脚本在任何时间都执行两个任务中的一个：要么显示添加记录表单，要么执行和添加一条新记录相关的SQL查询。

确定任务的逻辑从第4行开始，通过测试`$_POST`的值。如果`$_POST`超全局变量中没有值，用户必须提交表单，并且因此需要看到表单。在第6行到第89行，表单的HTML放入到了一个名为`$display_block`的字符串中。随后这个脚本跳出了`if...else`结构并且跳转到第189行，在那里输出HTML并且显示出`$display_block`的值，在这个例子中就是该表单。图20-2显示了其输出。

The screenshot shows a web browser window with the title 'Add an Entry' and the address bar displaying 'http://localhost/20/addentry.php'. The page content is a form titled 'Add an Entry' with the following fields and controls:

- First/Last Names:** Two adjacent text input fields.
- Street Address:** A single text input field.
- City/State/Zip:** Three adjacent text input fields.
- Address Type:** A group of three radio buttons labeled 'home', 'work', and 'other', with 'home' selected.
- Telephone Number:** A text input field followed by three radio buttons labeled 'home', 'work', and 'other', with 'home' selected.
- Fax Number:** A text input field followed by three radio buttons labeled 'home', 'work', and 'other', with 'home' selected.
- Email Address:** A text input field followed by three radio buttons labeled 'home', 'work', and 'other', with 'home' selected.
- Personal Note:** A large text area with a small icon in the bottom right corner.
- Add Entry:** A button at the bottom left of the form.

图20-2 添加记录表单

下面看看列表中带有if语句的代码，或者看看，如果表单提交了的话，会发生些什么，如程序清单20.3（续）所示。

程序清单20.3 名为addentry.php的基本记录添加脚本（续）

---

```
90: } else if ($_POST) {
91:     //time to add to tables, so check for required fields
92:     if (($_POST['f_name'] == "") || ($_POST['l_name'] == "")) {
93:         header("Location: addentry.php");
94:         exit;
95:     }
96:
```

```

97:      //connect to database
98:      doDB();
99:
100:     //create clean versions of input strings
101:     $safe_f_name = mysqli_real_escape_string($mysqli,
102:       $_POST['f_name']);
103:     $safe_l_name = mysqli_real_escape_string($mysqli,
104:       $_POST['l_name']);
105:     $safe_address = mysqli_real_escape_string($mysqli,
106:       $_POST['address']);
107:     $safe_city = mysqli_real_escape_string($mysqli,
108:       $_POST['city']);
109:     $safe_state = mysqli_real_escape_string($mysqli,
110:       $_POST['state']);
111:     $safe_zipcode = mysqli_real_escape_string($mysqli,
112:       $_POST['zipcode']);
113:     $safe_tel_number = mysqli_real_escape_string($mysqli,
114:       $_POST['tel_number']);
115:     $safe_fax_number = mysqli_real_escape_string($mysqli,
116:       $_POST['fax_number']);
117:     $safe_email = mysqli_real_escape_string($mysqli,
118:       $_POST['email']);
119:     $safe_note = mysqli_real_escape_string($mysqli,
120:       $_POST['note']);
121:
122:     //add to master_name table
123:     $add_master_sql = "INSERT INTO master_name (date_added,
124:       date_modified, f_name, l_name) VALUES
125:       (now(), now(), '". $safe_f_name."', '". $safe_l_name."')";
126:     $add_master_res = mysqli_query($mysqli, $add_master_sql)
127:       or die(mysqli_error($mysqli));
128:
129:     //get master_id for use with other tables
130:     $master_id = mysqli_insert_id($mysqli);
131:
132:     if (($_POST['address']) || ($_POST['city']) ||
133:       ($_POST['state']) || ($_POST['zipcode'])) {
134:       //something relevant, so add to address table
135:       $add_address_sql = "INSERT INTO address (master_id,
136:         date_added, date_modified, address, city, state,
137:         zipcode, type) VALUES
138:         ('". $master_id."', now(), now(),
139:         '". $safe_address."', '". $safe_city."',
140:         '". $safe_state."', '". $safe_zipcode."',
141:         '". $_POST['add_type']."'");
142:       $add_address_res = mysqli_query($mysqli, $add_address_sql)
143:         or die(mysqli_error($mysqli));
144:     }
145:
146:     if ($_POST['tel_number']) {
147:       //something relevant, so add to telephone table
148:       $add_tel_sql = "INSERT INTO telephone (master_id, date_added,
149:         date_modified, tel_number, type) VALUES
150:         ('". $master_id."', now(), now(),
151:         '". $safe_tel_number."', '". $_POST['tel_type']."'");
152:       $add_tel_res = mysqli_query($mysqli, $add_tel_sql)
153:         or die(mysqli_error($mysqli));
154:     }

```

```

155:
156:     if ($_POST['fax_number']) {
157:         //something relevant, so add to fax table
158:         $add_fax_sql = "INSERT INTO fax (master_id, date_added,
159:             date_modified, fax_number, type) VALUES
160:             ('".$master_id."', now(), now(), '".$safe_fax_number."',
161:             '".$_POST['fax_type']."'");
162:         $add_fax_res = mysqli_query($mysqli, $add_fax_sql)
163:             or die(mysqli_error($mysqli));
164:     }
165:     if ($_POST['email']) {
166:         //something relevant, so add to email table
167:         $add_email_sql = "INSERT INTO email (master_id, date_added,
168:             date_modified, email, type) VALUES
169:             ('".$master_id."', now(), now(), '".$safe_email."',
170:             '".$_POST['email_type']."'");
171:         $add_email_res = mysqli_query($mysqli, $add_email_sql)
172:             or die(mysqli_error($mysqli));
173:     }
174:
175:     if ($_POST['note']) {
176:         //something relevant, so add to notes table
177:         $add_notes_sql = "INSERT INTO personal_notes (master_id,
178:             date_added, date_modified, note) VALUES
179:             ('".$master_id."', now(), now(),
180:             '".$safe_note."'");
181:         $add_notes_res = mysqli_query($mysqli, $add_notes_sql)
182:             or die(mysqli_error($mysqli));
183:     }
184:     mysqli_close($mysqli);
185:     $display_block = "<p>Your entry has been added. Would you
186:         like to <a href=\"addentry.php\">add another</a>?</p>";
187: }
188: ?>
189: <!DOCTYPE html>
190: <head>
191: <title>Add an Entry</title>
192: </head>
193: <body>
194: <h1>Add an Entry</h1>
195: <?php echo $display_block; ?>
196: </body>
197: </html>

```

---

如果\$\_POST中有值的话，调用第90行的else条件，这意味着用户已经提交了表单。在这个简单的例子中，有两个字段被设置为必需字段，

即人物的名和姓。因此，第92行到第95行检查`$_POST ['f_name']`和`$_POST ['l_name']`中的值，并且如果任何一个值不存在的话就把用户重定向回到表单。

**提示：**

由于表单中的这两个必需字段标记为使用HTML 5的required属性，如果这两个字段中没有文本的话，表单将不会提交。然而，对于较早的浏览器，或者那些不支持INPUT字段中的HTML 5的required属性的设备，这一服务器端检测不会触发。

在检查过必需的字段之后，我们在第98行连接到数据库。一旦连接了数据库，可以安全地使用`mysqli_real_escape_string()`来检测用户输入，并且创建这些字符串的“安全”版本，以供INSERT语句使用。

接下来是多条插入语句，其中只有一条是必需的，就是向`master_name`插入一条记录。这发生在第123行到第127行。在插入之后，在第130行使用`mysqli_insert_id()`把这条记录的id提取了出来。我们在其余的SQL查询中使用了这个值，它现在被引用为`$master_id`。

把记录插入到其他表中的SQL查询都是有条件的，这意味着只有当某个条件为true的时候，它们才能发生。在第132行到第133行，我们看到必须满足的条件是，下列变量中的任何一个值存在：`$_POST ['address']`、`$_POST ['city']`、`$_POST ['state']`和`$_POST ['zipcode']`。如果条件满足，第135行到第143行创建并执行一个查询。

对于向`telephone`表（第146～154行）、`fax`表（第156～164行）、`email`表（第175～183行）以及`personal_notes`表（第123～130行）添加内容，同样的情况也成立。如果条件满足，记录将会插入到这些表中。

一旦通过了这组条件，给用户的消息也放置到`$display_block`变量中，脚本退出这个`if...else`结构并且在第189~197行显示HTML。图20-3展示了记录添加脚本的一个输出。

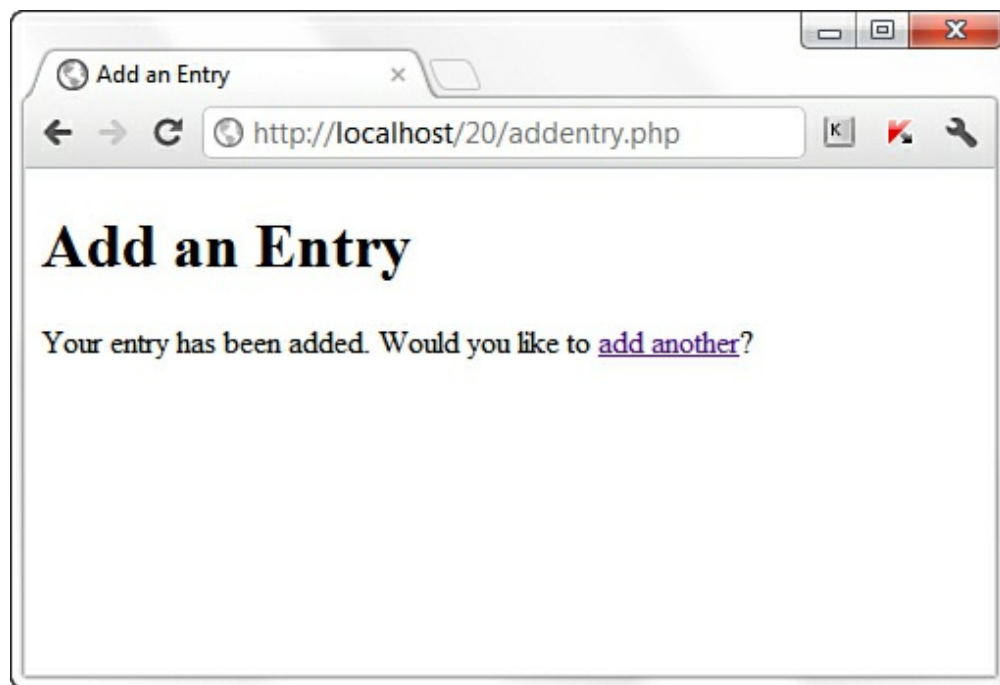


图20-3 已经添加了一条记录

使用这个表单添加几条记录，以便我们可以有一些值在下面的各节中使用。自己尝试修改这个脚本，以便在成功地插入记录后，把表单输入的值显示在屏幕上。



## 20.5 浏览记录

如果要通过MySQL监视器或者其他的界面来执行查询以验证在上一节中的工作，我们可能会对每个表都要输入SELECT \* FROM...感到厌烦。在本节中，我们将创建一个由两部分组成的脚本，向你展示如何从数据库选取和浏览记录。

程序清单20.4展示了名为selentry.php的选择和浏览的脚本，它拥有3部分：记录选择表单（第5行到第42行）和显示记录内容的代码（第43行到第172行），以及显示动态产生的字符串的HTML模板（第176行到第184行）。由于这段代码很长，我们将其分解为较小的代码段来讨论。

程序清单20.4 用来选取和浏览记录的名为selentry.php的脚本

---

```

1: <?php
2: include 'ch20_include.php';
3: doDB();
4:
5: if (!$_POST) {
6:     //haven't seen the selection form, so show it
7:     $display_block = "<h1>Select an Entry</h1>";
8:
9:     //get parts of records
10:    $get_list_sql = "SELECT id,
11:                    CONCAT_WS(' ', l_name, f_name) AS display_name
12:                    FROM master_name ORDER BY l_name, f_name";
13:    $get_list_res = mysqli_query($mysqli, $get_list_sql)
14:                    or die(mysqli_error($mysqli));
15:
16:    if (mysqli_num_rows($get_list_res) < 1) {
17:        //no records
18:        $display_block .= "<p><em>Sorry, no records to select!</em></p>";
19:
20:    } else {
21:        //has records, so get results and print in a form
22:        $display_block .= "
23:        <form method=\"post\" action=\"\".$_SERVER['PHP_SELF'].\">
24:        <p><label for=\"sel_id\">Select a Record:</label><br>
25:        <select id=\"sel_id\" name=\"sel_id\" required=\"required\">
26:        <option value=\"\">-- Select One --</option>";
27:
28:        while ($recs = mysqli_fetch_array($get_list_res)) {
29:            $id = $recs['id'];
30:            $display_name = stripslashes($recs['display_name']);
31:            $display_block .=
32:            "<option value=\"\".$id.\">\".$display_name.</option>";
33:        }
34:
35:        $display_block .= "
36:        </select>
37:        <button type=\"submit\" name=\"submit\"
38:        value=\"view\">View Selected Entry</button>
39:        </form>";
40:    }
41:    //free result
42:    mysqli_free_result($get_list_res);

```

和addentry.php脚本一样，selentry.php脚本在任何时候都执行两项任务中的一个：要么显示选择表单，要么执行与浏览记录相关的所有SQL查询。不管执行两个任务中的哪一个，都要用到数据库。因此，我们在

第2行导入了带有连接函数的文件，并且在第3行调用该函数。

确定任务的逻辑在第5行开始，通过测试超全局变量`$_POST`的值来实现。如果`$_POST`没有值，表示用户还没有看到选择表单，因此需要看到它。在第7行开始一个名为`$display_block`的字符串，并且这个字符串最终将存储组成记录选取表单的HTML。

在第10行到第12行，我们从`master_name`表的记录中选择了特定的字段来构建表单中的选择下拉选项。从这一步开始，我们只需要想要选择的记录的人的名字和ID。第16行测试了查询的结果，如果查询没有结果，我们不会构建一个表单。如果是这种情况，`$display_block`的值将会由一条错误消息填充，并且脚本将会结束，把最终的HTML显示到屏幕上。

然而，让我们假设在`master_name`表中有几条记录。在这个例子中，我们必须从查询结果中提取信息，以便能够构成表单。这在第28行到第33行完成，写入到`$display_block`字符串的表单元素都在这些行中。

我们在第42行停止了列表，但是，很快将看到第43行到这个脚本结束的内容。如果我们想要关闭这个if语句以及PHP代码块，并且在此时把`$display_block`的值显示到屏幕上，将会看到如图20-4所示的结果（可能具有不同的数据条目）。

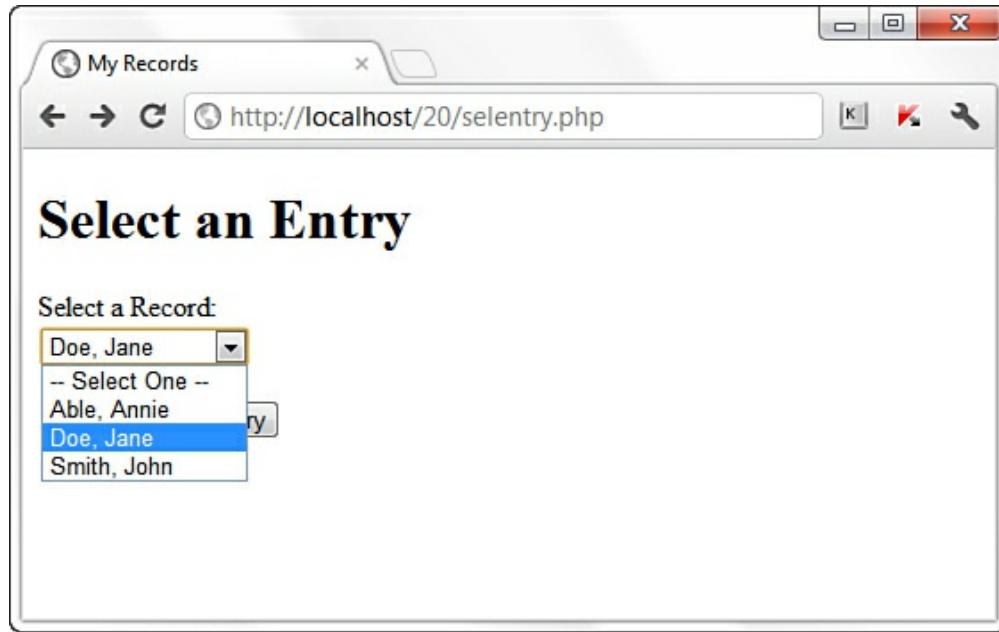


图20-4 记录选择表单

然而，我们必须完成selentry.php脚本，因此，我们从第43行开始继续程序清单20.4，从这里开始了if...else语句的else部分，如下所示。

程序清单20.4 用来选取和浏览记录的名为selentry.php的脚本（续）

---

```
43: } else if ($_POST) {
44:     //check for required fields
45:     if ($_POST['sel_id'] == "") {
46:         header("Location: selentry.php");
47:         exit;
48:     }
49:
50:     //create safe version of ID
51:     $safe_id = mysqli_real_escape_string($mysqli, $_POST['sel_id']);52:
52:
53:     //get master_info
54:     $get_master_sql = "SELECT concat_ws(' ',f_name,l_name) as display_name
55:                       FROM master_name WHERE id = '". $safe_id. "'";
56:     $get_master_res = mysqli_query($mysqli, $get_master_sql)
57:                       or die(mysqli_error($mysqli));
58:
59:     while ($name_info = mysqli_fetch_array($get_master_res)) {
60:         $display_name = stripslashes($name_info['display_name']);
61:     }
62:
63:     $display_block = "<h1>Showing Record for ". $display_name. "</h1>";
64:
65:     //free result
66:     mysqli_free_result($get_master_res);
67:
68:     //get all addresses
69:     $get_addresses_sql = "SELECT address, city, state, zipcode, type FROM
70:                          address WHERE master_id = '". $safe_id. "'";
71:     $get_addresses_res = mysqli_query($mysqli, $get_addresses_sql)
72:                          or die(mysqli_error($mysqli));
73:
74:     if (mysqli_num_rows($get_addresses_res) > 0) {
75:         $display_block .= "<p><strong>Addresses:</strong><br/>
76:         <ul>";
77:
78:         while ($add_info = mysqli_fetch_array($get_addresses_res)) {
79:             address = stripslashes($add_info['address']);
80:             $city = stripslashes($add_info['city']);
81:             $state = stripslashes($add_info['state']);
82:             $zipcode = stripslashes($add_info['zipcode']);
83:             $address_type = $add_info['type'];
84:
85:             $display_block .= "<li>$address $city $state $zipcode
86:                               ($address_type)</li>";
87:         }
88:         $display_block .= "</ul>";
89:     }
90:     //free result
91:     mysqli_free_result($get_addresses_res);
```

第43行包含了if...else语句的else部分，并且如果表单想要看一条特定的记录就调用这一部分。我们首先在第45行查看一个所需的字段，在这个例子中就是\$\_POST['sel\_id']的值。这个值把来自master\_name表的ID和在记录选择表单中做出的选择的ID进行比较。如果选择的ID不存在，用户将重定向到选择表单，当主键不存在的时候，我们无法很好地从一组表中收集信息。

假设有一个值赋给了\$\_POST["sel\_id"]，在第51行创建了一个安全版本。接下来，我们在第54行到第57行执行一个查询，它获取了我们想要浏览的记录的用户的名字。这个信息放置在我们现在已经熟悉的\$display\_block字符串中，随着脚本继续，这个字符串将继续构建。

第69行到第89行表示根据address表的一个查询结果构建最终的显示。如果所选择的个人在address表中没有记录，那就不会有什么内容添加到\$display\_block字符串。然而，如果有一个或多个条目，这个人的地址将会添加到\$display\_block字符串作为一个或多个未排序的列表元素，如第78行到第87行所示。

程序清单20.4的第92行到第168行执行相同类型的循环并且写入到\$display\_block变量，但是，操作的表不同。例如，第92行到第109行查看telephone表中的信息，并且创建相应的字符串以供添加到\$display\_block，如果存在任何信息的话。同样的结构在第114行到第130行针对fax表的信息而重复，在第135行到第150行针对email表的信息而重复，在第155行到第172行针对personal\_notes表中出现的信息而重复。

程序清单20.4 用来选取和浏览记录的名为selentry.php的脚本（续）

```
92: //get all tel
93: $get_tel_sql = "SELECT tel_number, type FROM telephone WHERE
94:                 master_id = '". $safe_id. "'";
95: $get_tel_res = mysqli_query($mysqli, $get_tel_sql)
96:                 or die(mysqli_error($mysqli));
97:
98: if (mysqli_num_rows($get_tel_res) > 0) {
99:     $display_block .= "<p><strong>Telephone:</strong><br/>
100:    <ul>";
101:
102:     while ($tel_info = mysqli_fetch_array($get_tel_res)) {
103:         $tel_number = stripslashes($tel_info['tel_number']);
104:         $tel_type = $tel_info['type'];
105:
106:         $display_block .= "<li>$tel_number ($tel_type)</li>";
107:     }
108:     $display_block .= "</ul>";
109: }
110: //free result
111: mysqli_free_result($get_tel_res);
112:
113: //get all fax
114: $get_fax_sql = "SELECT fax_number, type FROM fax WHERE
115:                 master_id = '". $safe_id. "'";
116: $get_fax_res = mysqli_query($mysqli, $get_fax_sql)
117:                 or die(mysqli_error($mysqli));
118:
119: if (mysqli_num_rows($get_fax_res) > 0) {
120:     $display_block .= "<p><strong>Fax:</strong><br/>
121:    <ul>";
122:
123:     while ($fax_info = mysqli_fetch_array($get_fax_res)) {
124:         $fax_number = stripslashes($fax_info['fax_number']);
125:         $fax_type = $fax_info['type'];
126:
127:         $display_block .= "<li>$fax_number ($fax_type)</li>";
128:     }
129:     $display_block .= "</ul>";
130: }
131: //free result
132: mysqli_free_result($get_fax_res);
133:
134: //get all email
135: $get_email_sql = "SELECT email, type FROM email WHERE
136:                   master_id = '". $safe_id. "'";
137: $get_email_res = mysqli_query($mysqli, $get_email_sql)
138:                   or die(mysqli_error($mysqli));
139: if (mysqli_num_rows($get_email_res) > 0) {
140:     $display_block .= "<p><strong>Email:</strong><br/>
141:    <ul>";
142:
143:     while ($email_info = mysqli_fetch_array($get_email_res)) {
144:         $email = stripslashes($email_info['email']);
145:         $email_type = $email_info['type'];
146:
147:         $display_block .= "<li>$email ($email_type)</li>";
```

```

148:         }
149:         $display_block .= "</ul>";
150:     }
151:     //free result
152:     mysqli_free_result($get_email_res);
153:
154:     //get personal note
155:     $get_notes_sql = "SELECT note FROM personal_notes WHERE
156:                     master_id = '". $safe_id . "'";
157:     $get_notes_res = mysqli_query($mysqli, $get_notes_sql)
158:                     or die(mysqli_error($mysqli));
159:
160:     if (mysqli_num_rows($get_notes_res) == 1) {
161:         while ($note_info = mysqli_fetch_array($get_notes_res)) {
162:             $note = nl2br(stripslashes($note_info['note']));
163:         }
164:         $display_block .= "<p><strong>Personal Notes:</strong><br/>
165:         $note</p>";
166:     }
167:     //free result
168:     mysqli_free_result($get_notes_res);

```

我们必须对脚本做一些清理和结束工作，如程序清单20.4的最后一部分所示。

程序清单20.4 用来选取和浏览记录的名为selentry.php的脚本（续）

---

```

169:     $display_block .= "<br/>
170:     <p style=\"text-align:center\">
171:     <a href=\"\".$_SERVER['PHP_SELF'].\"\">select another</a></p>";
172: }
173: //close connection to MySQL
174: mysqli_close($mysqli);
175: ?>
176: <!DOCTYPE html>
177: <html>
178: <head>
179: <title>My Records</title>
180: </head>
181: <body>
182: <?php echo $display_block; ?>
183: </body>
184: </html>

```

---



在第172行结束if...else，以及稍后结束PHP代码块之前，我们在第169行到第171行，简单地显示出回到选择表单的一个链接。第176行到脚本结束都是我们用来包含\$display\_block字符串的内容的HTML通用模板。

在从图20-4所示的表单选择一条记录之后，我们将会看到如图20-5所示的结果，当然，你的数据会有所不同。



图20-5 一条个人记录

当根据自己的记录自行尝试这个脚本的时候，只有针对那些拥有和他们相关联的附加数据的个体，才会看到信息。

例如，如果你的一个朋友有一个条目，并且你所拥有的只是在email表输入的一个Email地址，我们不会看到和地址、电话、传真或个人记录相关的其他任何文字，因为没有在这些表中输入相关的数据。

## 20.6 创建记录的删除机制

记录删除机制事实上和用来浏览一条记录的脚本相同。实际上，我们只要选取程序清单20.4中的前42行，将它们粘贴到一个名为delentry.php的新文件中，然后在第24行和第38行把“View”修改为“Delete”就可以了。

从第43行开始，delentry.php的代码的剩余部分如程序清单20.5所示。

程序清单20.5 用来选取和删除记录的名为delentry.php的脚本

---

```
43: } else if ($_POST) {
44:     //check for required fields
45:     if ($_POST['sel_id'] == "") {
46:         header("Location: delentry.php");
47:         exit;
48:     }
49:
50:     //create safe version of ID
51:     $safe_id = mysqli_real_escape_string($mysqli, $_POST['sel_id']);
52:
```

```

53: //issue queries
54: $del_master_sql = "DELETE FROM master_name WHERE
55:     id = '$safe_id.'";
56: $del_master_res = mysqli_query($mysqli, $del_master_sql)
57:     or die(mysqli_error($mysqli));
58:
59: $del_address_sql = "DELETE FROM address WHERE
60:     id = '$safe_id.'";
61: $del_address_res = mysqli_query($mysqli, $del_address_sql)
62:     or die(mysqli_error($mysqli));
63:
64: $del_tel_sql = "DELETE FROM telephone WHERE id = '$safe_id.'";
65: $del_tel_res = mysqli_query($mysqli, $del_tel_sql)
66:     or die(mysqli_error($mysqli));
67:
68: $del_fax_sql = "DELETE FROM fax WHERE id = '$safe_id.'";
69: $del_fax_res = mysqli_query($mysqli, $del_fax_sql)
70:     or die(mysqli_error($mysqli));
71:
72: $del_email_sql = "DELETE FROM email WHERE id = '$safe_id.'";
73: $del_email_res = mysqli_query($mysqli, $del_email_sql)
74:     or die(mysqli_error($mysqli));
75:
76: $del_note_sql = "DELETE FROM personal_notes WHERE
77:     id = '$safe_id.'";
78: $del_note_res = mysqli_query($mysqli, $del_note_sql)
79:     or die(mysqli_error($mysqli));
80:
81: mysqli_close($mysqli);
82:
83: $display_block = "<h1>Record(s) Deleted</h1>
84: <p>Would you like to
85: <a href='\".$_SERVER['PHP_SELF'].\">delete another</a>?</p>";
86: }
87: ?>
88: <!DOCTYPE html>
89: <html>
90: <head>
91: <title>My Records</title>
92: </head>
93: <body>
94: <?php echo $display_block; ?>
95: </body>
96: </html>

```

---

从第45行开始，脚本查找所需的字段\$\_POST ['sel\_id']，就像它在 selentry.php脚本中所做的那样。如果所需的值不存在，脚本将用户重定向到选择页面，但是，如果它存在，在第51行为其创建一个安全版本。

在第54行到第79行，查询从所有的表中删除所有和选择的个体相关的信息。第83行到第85行把一条消息放入\$display\_block中，脚本退出的时候把它以HTML显示到屏幕。图20-6显示了记录删除脚本的一个输出。

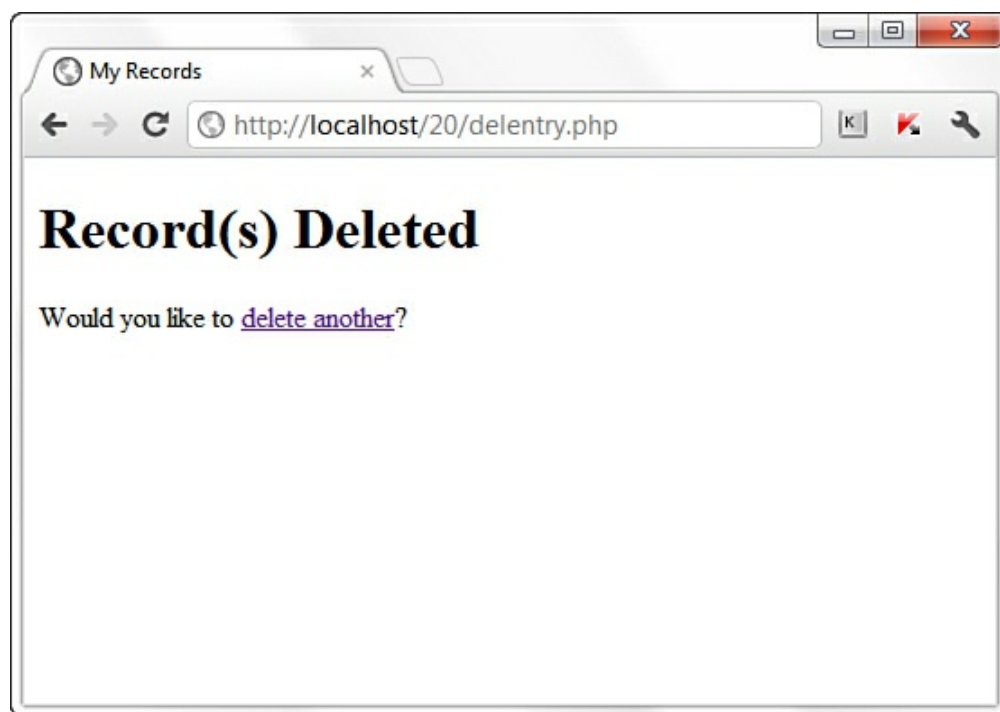


图20-6 删除一条记录

当删除一条记录之后，返回到记录选择页面，你将会注意到删除的个体不再存在于选择页面中，本来就该如此！

## 20.7 为一条记录添加子条目

本章至此，我们已经学习了如何添加、删除和浏览记录。还缺的就是当我们已经输入了一条记录之后，向相关的表添加附加的条目，例如，相对于工作电话号码的家庭电话号码。我们只需要对已有脚本略作改变就能做到这一点。

在程序清单20.4的selentr.php脚本中，把第185行到第186行修改为：

```
$display_block .= "<p style=\"text-align:center\">
<a href=\"addentry.php?master_id=\".$_POST['sel_id'].\"\">add info</a> ...
<a href=\"\".$_SERVER['PHP_SELF'].\"\">select another</a></p>";
```

这一修改只是向addentry.php脚本添加了一个链接，并且还通过\$\_GET['master\_id']传递一个可访问的变量。

现在，为了达到该目的，我们需要修改程序清单20.3中的addentry.php脚本。下面是对最初的脚本的修改的一个概括。

使用如下代码段替换最初的addentry.php的前10行。

```
<?php
include 'ch20_include.php';
doDB();

if ((!$_POST) || ($_GET['master_id'] != "")) {
    //haven't seen the form, so show it
    $display_block = "
    <form method=\"post\" action=\"\".$_SERVER['PHP_SELF'].\"\">;
    if (isset($_GET['master_id'])) {
```

```

//create safe version of ID
$safe_id = mysqli_real_escape_string($mysqli, $_GET['master_id']);

//get first, last names for display/tests validity
$get_names_sql = "SELECT concat_ws(' ', f_name, l_name) AS display_name
                  FROM master_name WHERE id = '". $safe_id. "'";
$get_names_res = mysqli_query($mysqli, $get_names_sql)
                  or die(mysqli_error($mysqli));

if (mysqli_num_rows($get_names_res) == 1) {
    while ($name_info = mysqli_fetch_array($get_names_res)) {
        $display_name = stripslashes($name_info['display_name']);
    }
}

if (isset($display_name)) {
    $display_block .= "<p>Adding information for
    <strong>$display_name</strong>:</p>";
} else {
    $display_block .= <<<END_OF_TEXT
    <fieldset>
    <legend>First/Last Names:</legend><br/>
    <input type="text" name="f_name" size="30"
        maxlength="75" required="required" />
    <input type="text" name="l_name" size="30"
        maxlength="75" required="required" />
    </fieldset>
END_OF_TEXT;
}
$display_block .= <<<END_OF_TEXT

<p><label for="address">Street Address:</label><br/>

```

这个代码段只是移动了表元素，只有当它们包含一条新的记录的时候才显示出姓和名字段。如果它们包含对一条记录的添加，个人的名字将从数据库中提取出来，为了美观也为了对ID进行有效性检查。

接下来，在最初的addentry.php脚本中找到如下的一行。

```
<button type="submit" name="submit" value="send">Add Entry</button>
```

直接在其上面添加如下内容。

```
END_OF_TEXT;
if ($_GET) {
    $display_block .= "<input type=\"hidden\" name=\"master_id\"
        value=\"".$_GET['master_id']."\">";
}

$display_block .= <<<END_OF_TEXT
```

这一修改确保了master\_id的值会传递给下一个任务，如果它存在。

注意最初脚本的第91行到第130行的内容，首先是把评论时间添加到表中，最后是获取\$master\_id的值。这些代码行应该用以下内容替代。



```

//time to add to tables, so check for required fields
if ((($_POST['f_name'] == "") || ($_POST['l_name'] == "")) &&
(!isset($_POST['master_id']))) {
    header("Location: addentry.php");
    exit;
}

//connect to database
doDB();

//create clean versions of input strings
$safe_f_name = mysqli_real_escape_string($mysqli,
    $_POST['f_name']);
$safe_l_name = mysqli_real_escape_string($mysqli,
    $_POST['l_name']);
$safe_address = mysqli_real_escape_string($mysqli,
    $_POST['address']);
$safe_city = mysqli_real_escape_string($mysqli,
    $_POST['city']);
$safe_state = mysqli_real_escape_string($mysqli,
    $_POST['state']);
$safe_zipcode = mysqli_real_escape_string($mysqli,
    $_POST['zipcode']);
$safe_tel_number = mysqli_real_escape_string($mysqli,
    $_POST['tel_number']);
$safe_fax_number = mysqli_real_escape_string($mysqli,
    $_POST['fax_number']);
$safe_email = mysqli_real_escape_string($mysqli,
    $_POST['email']);
$safe_note = mysqli_real_escape_string($mysqli,
    $_POST['note']);

if (!$_POST['master_id']) {
    //add to master_name table
    $add_master_sql = "INSERT INTO master_name (date_added, date_modified,
        f_name, l_name) VALUES (now(), now(),
        '". $safe_f_name . "', '". $safe_l_name . "')";
    $add_master_res = mysqli_query($mysqli, $add_master_sql)
        or die(mysqli_error($mysqli));

    //get master_id for use with other tables
    $master_id = mysqli_insert_id($mysqli);
} else {
    $master_id = mysqli_real_escape_string($mysqli, $_POST['master_id']);
}

```

这些代码修改了对所需字段的检查，允许脚本在没有姓和名的情况下继续，但是，只有当它有一个\$\_POST ['master\_id']值时才可以。

然后，脚本连接到数据库来执行我们希望它执行的所有添加，但是，如果\$\_POST ['master\_id']的值已经存在的话，它会略过对master\_name表的添加。

最后，在脚本的处理插入到personal\_notes表的SQL语句部分，把INSERT改为UPDATE来处理notes字段的更新。

```
$add_notes_sql = "UPDATE personal_notes set note = '". $safe_note."',  
                 date_modified = now() WHERE master_id = '". $master_id."'";
```

新的脚本应该如程序清单20.6所示。

程序清单20.6 新的addentry.php脚本

---

```

1: <?php
2: include 'ch20_include.php';
3: doDB();
4:
5: if ((!$_POST) || ($_GET['master_id'] != "")) {
6:     //haven't seen the form, so show it
7:     $display_block = "
8:     <form method=\"post\" action=\"\".$_SERVER['PHP_SELF'].\">";
9:     if (isset($_GET['master_id'])) {
10:         //create safe version of ID
11:         $safe_id = mysqli_real_escape_string($mysqli, $_GET['master_id']);
12:
13:         //get first, last names for display/tests validity
14:         $get_names_sql = "SELECT concat_ws(' ', f_name, l_name) AS
            display_name
15:                             FROM master_name WHERE id = '\".$safe_id.\"'";
16:         $get_names_res = mysqli_query($mysqli, $get_names_sql)
17:                             or die(mysqli_error($mysqli));
18:
19:         if (mysqli_num_rows($get_names_res) == 1) {
20:             while ($name_info = mysqli_fetch_array($get_names_res)) {
21:                 $display_name = stripslashes($name_info['display_name']);
22:             }
23:         }
24:     }
25:
26:     if (isset($display_name)) {
27:         $display_block .= "<p>Adding information for
28:         <strong>$display_name</strong>:</p>";
29:     } else {
30:         $display_block .= <<<END_OF_TEXT          <fieldset>
31:         <legend>First/Last Names:</legend><br/>
32:         <input type="text" name="f_name" size="30"
33:             maxlength="75" required="required" />
34:         <input type="text" name="l_name" size="30"
35:             maxlength="75" required="required" />
36:         </fieldset>
37:     END_OF_TEXT;
38:     }
39:     $display_block .= <<<END_OF_TEXT
40:     <p><label for="address">Street Address:</label><br/>
41:     <input type="text" id="address" name="address"
42:         size="30" /></p>
43:
44:     <fieldset>

```

```

45:     <legend>City/State/Zip:</legend><br/>
46:     <input type="text" name="city" size="30" maxlength="50" />
47:     <input type="text" name="state" size="5" maxlength="2" />
48:     <input type="text" name="zipcode" size="10" maxlength="10" />
49: </fieldset>
50:
51: <fieldset>
52:     <legend>Address Type:</legend><br/>
53:     <input type="radio" id="add_type_h" name="add_type"
54:         value="home" checked />
55:         <label for="add_type_h">home</label>
56:     <input type="radio" id="add_type_w" name="add_type"
57:         value="work" />
58:         <label for="add_type_w">work</label>
59:     <input type="radio" id="add_type_o" name="add_type"
60:         value="other" />
61:         <label for="add_type_o">other</label>
62: </fieldset>
63:
64: <fieldset>
65:
66:     <legend>Telephone Number:</legend><br/>
67:     <input type="text" name="tel_number" size="30" maxlength="25" />
68:     <input type="radio" id="tel_type_h" name="tel_type"
69:         value="home" checked />
70:         <label for="tel_type_h">home</label>
71:     <input type="radio" id="tel_type_w" name="tel_type"
72:         value="work" />
73:         <label for="tel_type_w">work</label>
74:     <input type="radio" id="tel_type_o" name="tel_type"
75:         value="other" />
76:         <label for="tel_type_o">other</label>
77: </fieldset>
78:
79: <fieldset>
80:     <legend>Fax Number:</legend><br/>
81:     <input type="text" name="fax_number" size="30" maxlength="25" />
82:     <input type="radio" id="fax_type_h" name="fax_type"
83:         value="home" checked />
84:         <label for="fax_type_h">home</label>
85:     <input type="radio" id="fax_type_w" name="fax_type"
86:         value="work" />
87:         <label for="fax_type_w">work</label>
88:     <input type="radio" id="fax_type_o" name="fax_type"
89:         value="other" />
90:         <label for="fax_type_o">other</label>
91: </fieldset>
92:
93: <fieldset>
94:     <legend>Email Address:</legend><br/>
95:     <input type="email" name="email" size="30" maxlength="150" />
96:     <input type="radio" id="email_type_h" name="email_type"
97:         value="home" checked />
98:         <label for="email_type_h">home</label>
99:     <input type="radio" id="email_type_w" name="email_type"
100:         value="work" />
101:         <label for="email_type_w">work</label>

```



```

102:     <input type="radio" id="email_type_o" name="email_type"
103:         value="other" />
104:         <label for="email_type_o">other</label>
105:     </fieldset>
106:
107:     <p><label for="note">Personal Note:</label><br/>
108:     <textarea id="note" name="note" cols="35"
109:         rows="3"></textarea></p>
110: END_OF_TEXT;
111:     if ($_GET) {
112:         $display_block .= "<input type=\"hidden\" name=\"master_id\"
113:             value=\"".$_GET['master_id']."\">";
114:     }
115:     $display_block .= <<<END_OF_TEXT
116:     <button type="submit" name="submit"
117:         value="send">Add Entry</button>
118:     </form>
119: END_OF_TEXT;
120: } else if ($_POST) {
121:     //time to add to tables, so check for required fields
122:     if ((($_POST['f_name'] == "") || ($_POST['l_name'] == "")) &&
123:         (!isset($_POST['master_id']))) {
124:         header("Location: addentry.php");
125:         exit;
126:     }
127:
128:     //connect to database
129:     doDB();
130:     //create clean versions of input strings
131:     $safe_f_name = mysqli_real_escape_string($mysqli,
132:         $_POST['f_name']);
133:     $safe_l_name = mysqli_real_escape_string($mysqli,
134:         $_POST['l_name']);
135:     $safe_address = mysqli_real_escape_string($mysqli,
136:         $_POST['address']);
137:     $safe_city = mysqli_real_escape_string($mysqli,
138:         $_POST['city']);
139:     $safe_state = mysqli_real_escape_string($mysqli,
140:         $_POST['state']);
141:     $safe_zipcode = mysqli_real_escape_string($mysqli,
142:         $_POST['zipcode']);
143:     $safe_tel_number = mysqli_real_escape_string($mysqli,
144:         $_POST['tel_number']);
145:     $safe_fax_number = mysqli_real_escape_string($mysqli,
146:         $_POST['fax_number']);
147:     $safe_email = mysqli_real_escape_string($mysqli,
148:         $_POST['email']);
149:     $safe_note = mysqli_real_escape_string($mysqli,
150:         $_POST['note']);
151:
152:     if (!$_POST['master_id']) {
153:         //add to master_name table
154:         $add_master_sql = "INSERT INTO master_name (date_added,
155:             date_modified,
156:             f_name, l_name) VALUES (now(), now(),
157:             '".$_safe_f_name."', '".$_safe_l_name."')";
158:         $add_master_res = mysqli_query($mysqli, $add_master_sql)

```



```

158:                                     or die(mysql_error($mysqli));
159:
160:         //get master_id for use with other tables
161:         $master_id = mysqli_insert_id($mysqli);
162:     } else {
163:         $master_id = mysqli_real_escape_string($mysqli,
            $_POST['master_id']);
164:     }
165:
166:     if (($_POST['address']) || ($_POST['city']) ||
167:         ($_POST['state']) || ($_POST['zipcode'])) {
168:         //something relevant, so add to address table
169:         $add_address_sql = "INSERT INTO address (master_id,
170:             date_added, date_modified, address, city, state,
171:             zipcode, type) VALUES
172:             ('".$master_id."', now(), now(),
173:             '".$safe_address."', '".$safe_city."',
174:             '".$safe_state."', '".$safe_zipcode."',
175:             '".$_POST['add_type']. "')";
176:         $add_address_res = mysqli_query($mysqli, $add_address_sql)
177:             or die(mysql_error($mysqli));
178:     }
179:
180:     if ($_POST['tel_number']) {
181:         //something relevant, so add to telephone table
182:         $add_tel_sql = "INSERT INTO telephone (master_id, date_added,
183:             date_modified, tel_number, type) VALUES
184:             ('".$master_id."', now(), now(),
185:             '".$safe_tel_number."', '".$_POST['tel_type']. "')";
186:         $add_tel_res = mysqli_query($mysqli, $add_tel_sql)
187:             or die(mysql_error($mysqli));
188:     }
189:
190:     if ($_POST['fax_number']) {
191:         //something relevant, so add to fax table
192:         $add_fax_sql = "INSERT INTO fax (master_id, date_added,
193:             date_modified, fax_number, type) VALUES
194:             ('".$master_id."', now(), now(), '".$safe_fax_number."',
195:             '".$_POST['fax_type']. "')";
196:         $add_fax_res = mysqli_query($mysqli, $add_fax_sql)
197:             or die(mysql_error($mysqli));
198:     }
199:     if ($_POST['email']) {
200:         //something relevant, so add to email table
201:         $add_email_sql = "INSERT INTO email (master_id, date_added,
202:             date_modified, email, type) VALUES
203:             ('".$master_id."', now(), now(), '".$safe_email."',
204:             '".$_POST['email_type']. "')";
205:         $add_email_res = mysqli_query($mysqli, $add_email_sql)
206:             or die(mysql_error($mysqli));
207:     }
208:
209:     if ($_POST['note']) {
210:         //something relevant, so add to notes table
211:         $add_notes_sql = "UPDATE personal_notes set note =
212:             '".$safe_note."', date_modified = now()

```



```
213:             WHERE master_id = '". $master_id. "'";
214:     }
215:     mysqli_close($mysqli);
216:     $display_block = "<p>Your entry has been added.  Would you
217:         like to <a href=\"addentry.php\">add another</a>?</p>";
218: }
219: ?>
220: <!DOCTYPE html>
221: <head>
222: <title>Add an Entry</title>
223: </head>
224: <body>
225: <h1>Add an Entry</h1>
226: <?php echo $display_block; ?>
227: </body>
228: </html>
```

---

可以选择一条记录来浏览并且点击add info链接来测试这个修改后的脚本。应该会看到如图20-7所示的一个页面。

The screenshot shows a web browser window with the title 'Add an Entry' and the URL 'http://localhost/20/addentry2.php?master\_id=3'. The page content is as follows:

## Add an Entry

Adding information for **John Smith**:

Street Address:

City/State/Zip:

Address Type:  
☒ home ☐ work ☐ other

Telephone Number:  
 ☒ home ☐ work ☐ other

Fax Number:  
 ☒ home ☐ work ☐ other

Email Address:  
 ☒ home ☐ work ☐ other

Personal Note:

图20-7 添加记录

在提交了这个表单后，我们可以回到选择过程并且查看记录来验证我们已经做出的改变。

## 20.8 小结

在这个动手实践的一章中，我们应用基本的PHP和MySQL知识来创建了一个个人地址簿。我们学习了如何创建数据库表以及进行记录添加、删除和简单浏览的脚本。我们还学习了添加附加到一条主条目的多条记录的过程。

## 20.9 Q&A

**Q:** 如果我想在地址簿中添加额外的字段，例如为一个人的生日或其他信息添加记录，该怎么做？

**A:** 不同的表用于地址、电话、传真、Email和个人信息，因为一个人可能有多条包含这些信息的记录。对于一个人的生日来说，一个人只能有一条这样的信息，所以一个关系数据库可能有些牛刀杀鸡，因为每个存在的用户只有一条记录。因此，要添加一个人的生日，我们应该为master\_name表添加一个字段。在为该表添加其他信息的情况下，首先要看一看一个人是只有该信息的一个实例（例如生日），还是有多个实例（例如Email地址）。如果是后者，创建一个像address、telephone、fax、email或personal\_notes这样的表，使用master\_id作为外键。

## 20.10 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 当通过查询字符串传递一个变量的时候，它应该属于哪个超全局变量？
2. 对于master\_name表中的每一条单个的记录，在address、email、telephone和fax表中可以有多少条记录？
3. 通过哪个数据库字段将额外的记录附加到主记录？

## 解答

1. \$\_GET超全局变量
2. 只要你愿意，可以有任意多条，这就是关系！
3. master\_id字段

## 思考题

1. 浏览每个管理员脚本并且修改代码，以便在每个屏幕的底部显示一个到菜单的链接。
2. 使用addentry.php脚本的第二个版本来向数据库中的记录添加二级联络信息。图20-8展示当二级联络信息添加给一条记录之后，它是如何显示的。



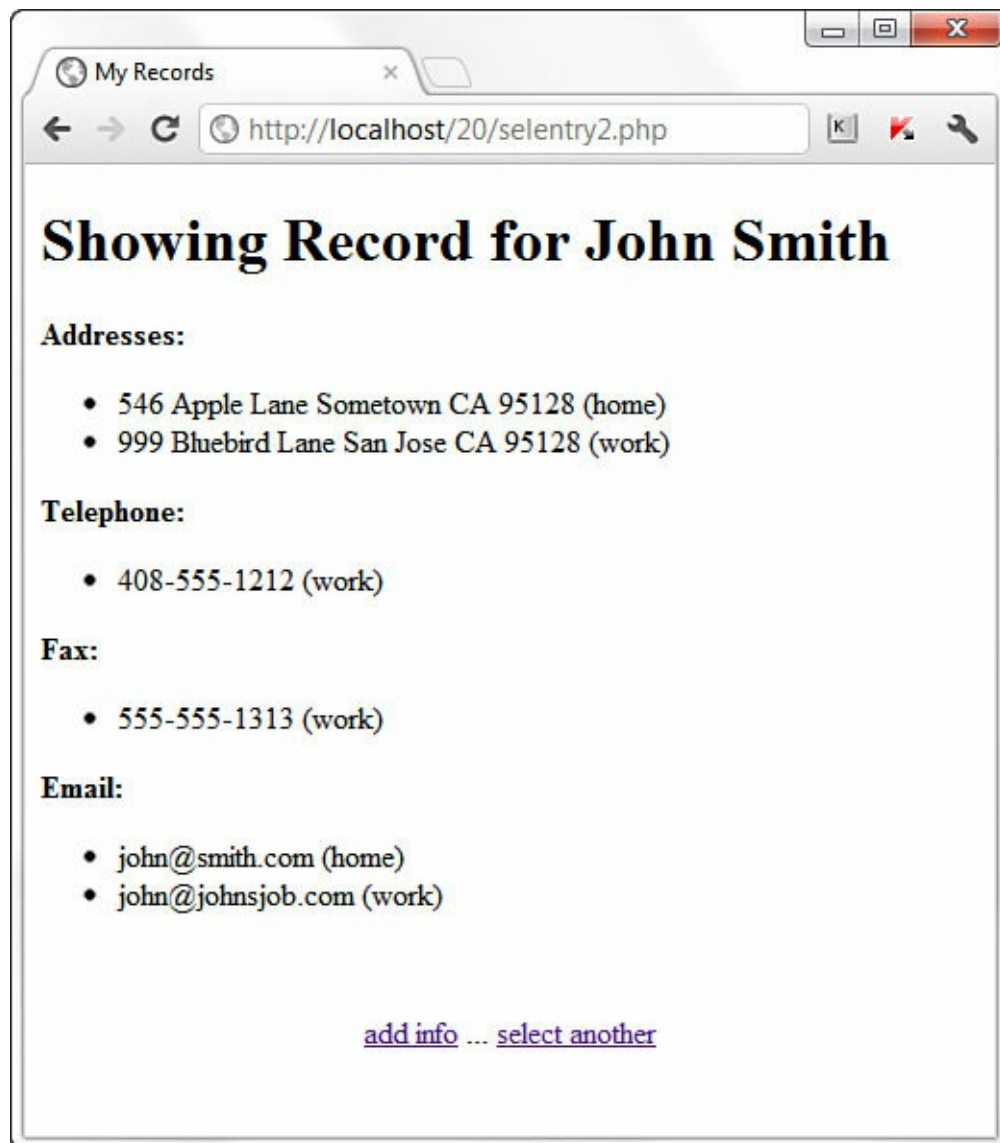


图20-8 表中带有多条目的一个单条记录

## 第21章 创建一个简单的讨论论坛

在本章中，你将学习

- 如何为一个简单的讨论论坛创建表格。
- 如何为一个简单的讨论论坛创建输入表单。
- 如何显示一个简单的讨论论坛。

在本章中，我们将要学习一个简单的讨论论坛背后的设计过程。这包括开发数据库表、用户输入表单以及显示结果。当我们像这样分解讨论论坛的时候，这样的任务看上去好像很简单，并且实际上它确实简单。最终目标是理解开发一个像讨论论坛这样的东西所需的概念和关系，而不是要创建世界上功能最完善的系统，实际上，你将看到它并不是功能非常完善，但它真的是有关系的。

## 21.1 设计数据库表

考虑一个论坛的基本组成部分：主题和帖子。如果论坛的创立者使用正确的话，一个论坛应该包括几个主题，并且每个主题应该有用户所提交的一个或多个帖子。了解了这一点，我们就应该意识到，帖子是通过一个键字段联系到主题的。这个键构成了两个表之间的关系。

考虑一下主题本身的需求。我们肯定需要一个字段来保存标题，并且随后可能需要字段来保存创建时间和创建这个主题的用户身份。类似的，考虑一下帖子的需求：我们需要存储帖子的文本、创建的时间以及创建者。最重要的是，我们需要将帖子绑定到主题的键。

下面的两个表创建语句创建了两个表，这两个表叫做forum\_topics和forum\_posts。

```
CREATE TABLE forum_topics (  
    topic_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    topic_title VARCHAR (150),  
    topic_create_time DATETIME,  
    topic_owner VARCHAR (150)  
);  
CREATE TABLE forum_posts (  
    post_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    topic_id INT NOT NULL,  
    post_text TEXT,  
    post_create_time DATETIME,  
    post_owner VARCHAR (150)  
);
```

### 提示：

在这个简单的论坛示例中，我们将通过用户的Email地址来识别用户，而不需要任何其他类型的登录标识符。

---

现在，我们应该有两个空的表等待输入。在下一节中，我们将创建输入表单用来添加一个主题和一个帖子。

## 21.2 为共同函数创建一个包含文件

前面两章为共同函数创建了一个包含文件，使得脚本更为精简，并且帮助管理可能随着时间而变化的信息，例如数据库用户名和密码。在本章中也是这样。程序清单21.1包含了本章中的脚本共享的代码。

程序清单21.1 包含文件中的共同函数

---

```
1: <?php
2: function doDB() {
3:     global $mysqli;
4:
5:     //connect to server and select database; you may need it
6:     $mysqli = mysqli_connect("localhost", "joeuser",
7:         "somepass", "testDB");
8:
9:     //if connection fails, stop script execution
10:    if (mysqli_connect_errno()) {
11:        printf("Connect failed: %s\n", mysqli_connect_error());
12:        exit();
13:    }
14: }
15: ?>
```

---

第2行到第14行建立了一个数据库连接函数doDB()。如果没有成功建立连接，在调用这个函数的时候，脚本将会退出；否则，它将让\$mysqli的值可供脚本其他部分使用。

把这个文件保存为ch21\_include.php并将其放置到Web服务器上。本章中的其他代码将会在脚本的前几行中包含这个文件。

## 21.3 创建输入表单和脚本

在我们可以添加任何帖子之前，我们必须向论坛添加一个主题。同时添加一个主题及该主题的第一个帖子，是论坛创建过程中常见的做法。从一个用户的观点来看，添加一个主题然后返回，然后选择该主题并添加一个回复，这么做并没有多少意义。我们希望过程尽可能地平顺。程序清单21.2给出了一个创建新主题的表单，其中包含了为主题中的第一个帖子留出的空间。

程序清单21.2 添加一个主题的表单

---

```
1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>Add a Topic</title>
5: </head>
6: <body>
7: <h1>Add a Topic</h1>
8: <form method="post" action="do_addtopic.php">
9:
10: <p><label for="topic_owner">Your Email Address:</label><br/>
11: <input type="email" id="topic_owner" name="topic_owner" size="40"
12:     maxlength="150" required="required" /></p>
13:
14: <p><label for="topic_title">Topic Title:</label><br/>
15: <input type="text" id="topic_title" name="topic_title" size="40"
16:     maxlength="150" required="required" /></p>
17: <p><label for="post_text">Post Text:</label><br/>
18: <textarea id="post_text" name="post_text" rows="8"
19:     cols="40" ></textarea></p>
20:
21: <button type="submit" name="submit" value="submit">Add Topic</button>
22:
23: </form>
24: </body>
25: </html>
```

---

看上去很简单，我们可以从图21-1看到，表单中出现了3个字段，

这是我们需要在各个表中完成的，你的脚本和数据库可以填写其他的内容。把程序清单21.2保存为类似addtopic.html这样的文件并将其放置到Web服务器文档根目录下，以便我们可以执行。

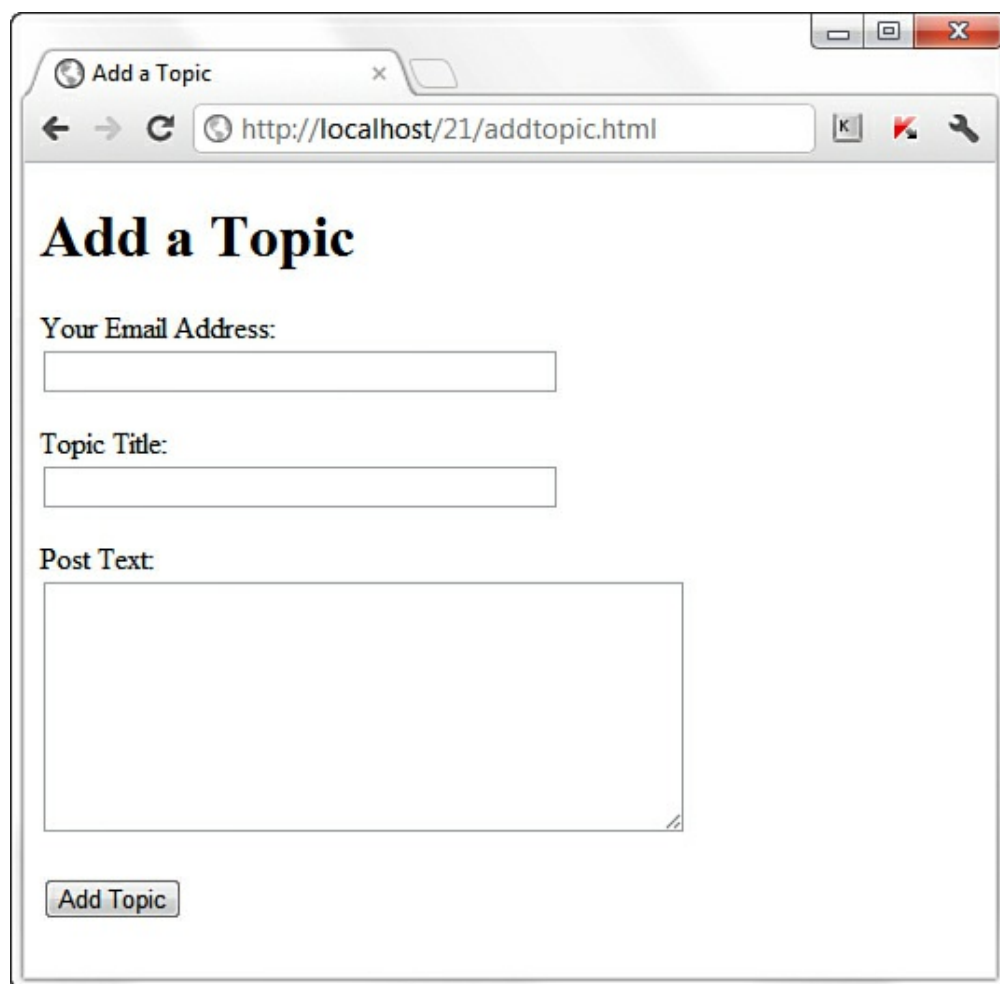
A screenshot of a web browser window. The browser's address bar shows 'http://localhost/21/addtopic.html'. The page title is 'Add a Topic'. The main content area contains a form with the following elements: a heading 'Add a Topic' in a large, bold, serif font; a label 'Your Email Address:' followed by a single-line text input field; a label 'Topic Title:' followed by a single-line text input field; a label 'Post Text:' followed by a multi-line text area; and a button labeled 'Add Topic' at the bottom left.

图21-1 创建主题的表单

要在forum\_topics表中创建条目，可以使用来自输入表单的变量\$\_POST['topic\_title']和\$\_POST['topic\_owner']中的值。topic\_id和topic\_create\_time字段分别自动增加或通过MySQL函数now()添加。

类似地，在forum\_posts表中，我们使用来自输入表单的

\$\_POST['post\_text']和\$\_POST ['topic\_owner']中的值，而post\_id、post\_create\_time和topic\_id字段将会自动增加，或被提供。由于我们需要topic\_id字段的一个值，从而完成forum\_posts表中的条目，我们知道，这个查询必须在向forum\_topics表插入记录的查询之后执行。程序清单21.3创建了一个脚本来向表添加这些记录。

程序清单21.3 添加一个主题的脚本

---

```
1:  <?php
2:  include 'ch21_include.php';
3:  doDB();
4:
5:  //check for required fields from the form
6:  if ((!$_POST['topic_owner']) || (!$_POST['topic_title']) ||
7:      (!$_POST['post_text'])) {
8:      header("Location: addtopic.html");
9:      exit;
10: }
11:
12: //create safe values for input into the database
13: $clean_topic_owner = mysqli_real_escape_string($mysqli,
14:         $_POST['topic_owner']);
15: $clean_topic_title = mysqli_real_escape_string($mysqli,
```



```

16:             $_POST['topic_title']);
17: $clean_post_text = mysqli_real_escape_string($mysqli,
18:             $_POST['post_text']);
19:
20: //create and issue the first query
21: $add_topic_sql = "INSERT INTO forum_topics
22:             (topic_title, topic_create_time, topic_owner)
23:             VALUES ('".$_clean_topic_title."', now(),
24:             '".$_$clean_topic_owner."')";
25:
26: $add_topic_res = mysqli_query($mysqli, $add_topic_sql)
27:             or die(mysqli_error($mysqli));
28:
29: //get the id of the last query
30: $topic_id = mysqli_insert_id($mysqli);
31:
32: //create and issue the second query
33: $add_post_sql = "INSERT INTO forum_posts
34:             (topic_id, post_text, post_create_time, post_owner)
35:             VALUES ('".$topic_id."', '".$_clean_post_text."',
36:             now(), '".$_clean_topic_owner."')";
37:
38: $add_post_res = mysqli_query($mysqli, $add_post_sql)
39:             or die(mysqli_error($mysqli));
40: //close connection to MySQL
41: mysqli_close($mysqli);
42:
43: //create nice message for user
44: $display_block = "<p>The <strong>".$_POST["topic_title"]."</strong>
45:     topic has been created.</p>";
46: ?>
47: <!DOCTYPE html>
48: <html>
49: <head>
50: <title>New Topic Added</title>
51: </head>
52: <body>
53: <h1>New Topic Added</h1>
54: <?php echo $display_block; ?>
55: </body>
56: </html>

```

---

第2行到第3行包含了用户创建的函数的文件，并调用了数据库连接函数。接着，第6行到第10行检查要完成两个表都必需的3个字段，topic owner、topic title以及帖子的一些文本。如果这些字段中的任何一个不存在，用户将被重定向到最初的表单。第13行到第18行创建了这些变量

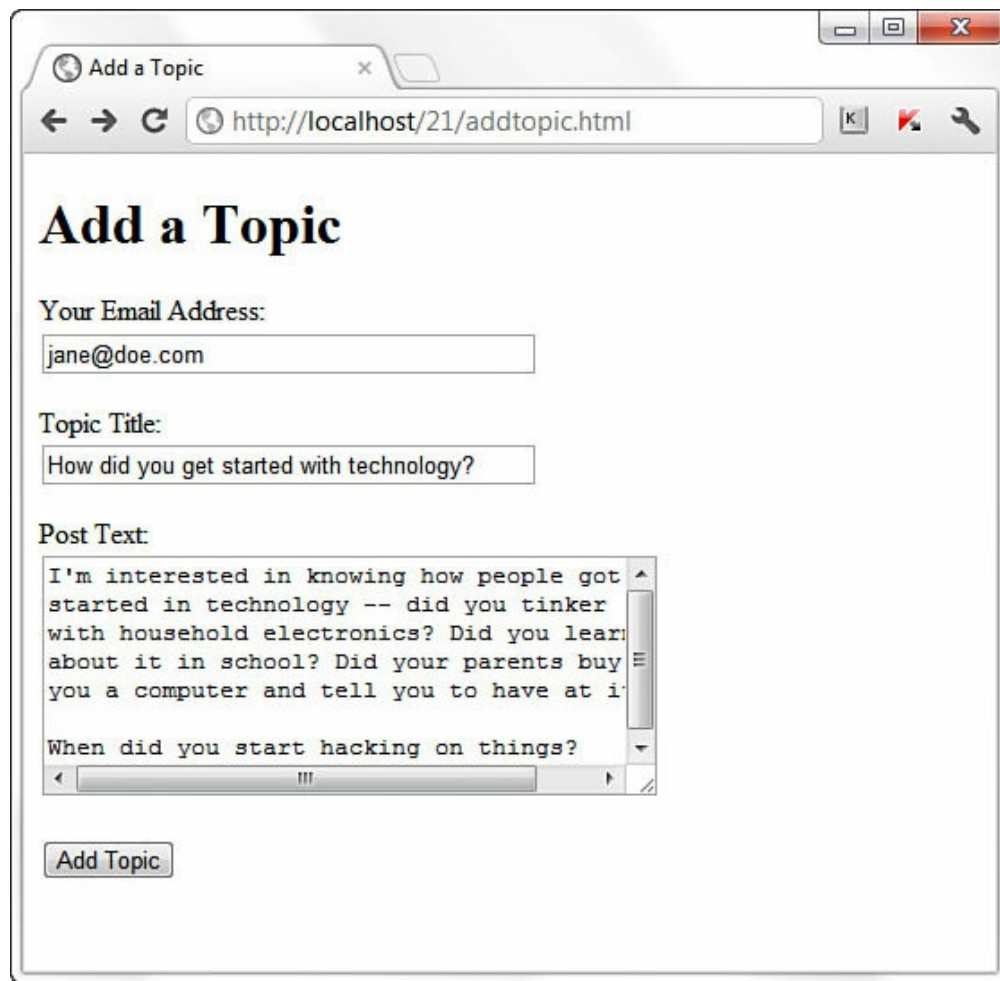
内容的数据库安全版本。

第21行到第27行创建并插入了第一个记录，它向forum\_topics表添加了主题。注意，第一个字段保持空白，以便自动增加的值可以由每个表定义的系统来添加。MySQL的函数now()用来在插入的时候记录下当前时间的戳。记录中的其他字段使用来自表单的值填充。

第30行展示了一个方便的函数mysqli\_insert\_id()的使用。这个函数可以从这个脚本插入到数据库中的最后一条记录中获取主键ID。在这个例子中，mysqli\_insert\_id()从forum\_topics表获取了id值，这将成为forum\_posts表中的topic\_id字段的值。

第33行到第39行创建并插入了第二个查询，再次使用了已知信息以及系统提供信息的混合。第二个查询把用户帖子的文本添加到了forum\_posts表。第44行到第45行为用户创建了一个显示字符串，并且脚本剩下的内容负责完成浏览器所要显示的HTML。

把这个程序清单保存为do\_addtopic.php（即前面的脚本所做的动作的名称）并且将其放置到Web服务器的文档根目录下。完成图21-1创建的表单，然后提交，我们应该看到New Topic Added消息。图21-2和21-3显示了事件顺序。



A screenshot of a web browser window titled "Add a Topic". The address bar shows "http://localhost/21/addtopic.html". The page content includes a title "Add a Topic", a form for "Your Email Address" with the value "jane@doe.com", a form for "Topic Title" with the value "How did you get started with technology?", and a form for "Post Text" with the value "I'm interested in knowing how people got started in technology -- did you tinker with household electronics? Did you learn about it in school? Did your parents buy you a computer and tell you to have at it? When did you start hacking on things?". There is an "Add Topic" button at the bottom.

Add a Topic

http://localhost/21/addtopic.html

## Add a Topic

Your Email Address:

jane@doe.com

Topic Title:

How did you get started with technology?

Post Text:

I'm interested in knowing how people got started in technology -- did you tinker with household electronics? Did you learn about it in school? Did your parents buy you a computer and tell you to have at it? When did you start hacking on things?

Add Topic

图21-2 添加一个主题以及第一条帖子

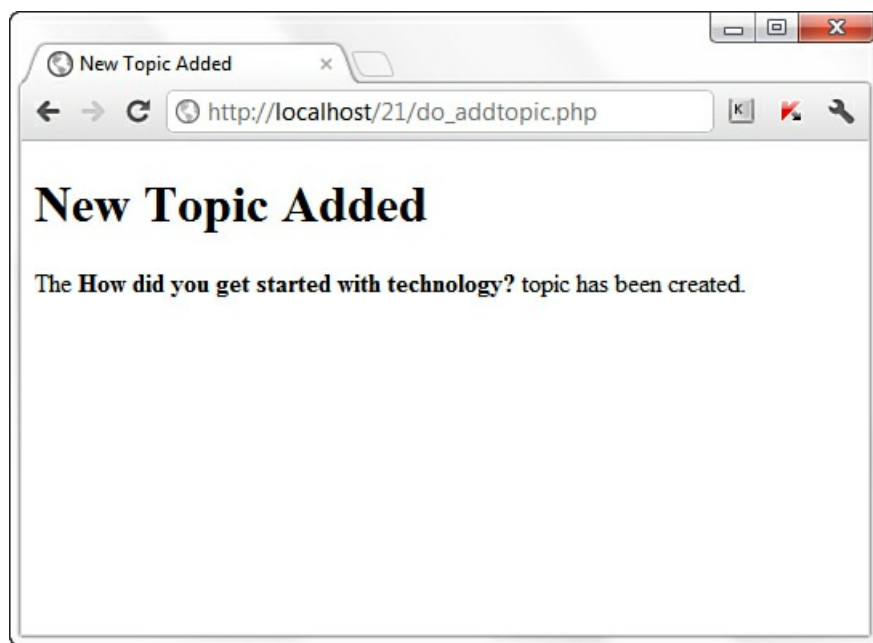


图21-3 一个主题和第一条帖子的成功添加

在下一节中，我们将会把另外两个神秘的部分组合到一起：显示主题和帖子，并且回复一个主题。

## 21.4 显示主题列表

既然在数据库里有了一个主题以及至少一个帖子，我们可以显示这一信息并且让人们添加新的主题或者回复已有的帖子。在程序清单21.4中，我们退回一步并且创建一个页面列出论坛中的所有主题。这个页面显示了每个主题的基本信息并且给用户提供一个添加新主题的连接，前文已经创建了添加新主题的表单和脚本。程序清单21.4中的代码代表了论坛的一个入口页面。

尽管程序清单21.4看上去有很多代码，它实际上只涉及到我们已经遇到过的很多小的、简单的概念，从第2行到第3行中的include()函数和数据库连接函数开始。

程序清单21.4 主题列表脚本

---

```
1: <?php
2: include 'ch21_include.php';
3: doDB();
4:
5: //gather the topics
6: $get_topics_sql = "SELECT topic_id, topic_title,
7:                     DATE_FORMAT(topic_create_time, '%b %e %Y at %r') AS
8:                     fmt_topic_create_time, topic_owner FROM forum_topics
9:                     ORDER BY topic_create_time DESC";
10: $get_topics_res = mysqli_query($mysqli, $get_topics_sql)
11:     or die(mysqli_error($mysqli));
12:
13: if (mysqli_num_rows($get_topics_res) < 1) {
14:     //there are no topics, so say so
15:     $display_block = "<p><em>No topics exist.</em></p>";
16: } else {
17:     //create the display string
18:     $display_block <<<END_OF_TEXT
19:     <table>
20:     <tr>
21:     <th>TOPIC TITLE</th>
22:     <th># of POSTS</th>
23:     </tr>
24: END_OF_TEXT;
25:
26:     while ($topic_info = mysqli_fetch_array($get_topics_res)) {
27:         $topic_id = $topic_info['topic_id'];
28:         $topic_title = stripslashes($topic_info['topic_title']);
29:         $topic_create_time = $topic_info['fmt_topic_create_time'];
30:         $topic_owner = stripslashes($topic_info['topic_owner']);
31:
32:         //get number of posts
33:         $get_num_posts_sql = "SELECT COUNT(post_id) AS post_count FROM
34:                             forum_posts WHERE topic_id = '". $topic_id. "'";
```

```

35:         $get_num_posts_res = mysqli_query($mysqli, $get_num_posts_sql)
36:             or die(mysqli_error($mysqli));
37:
38:         while ($posts_info = mysqli_fetch_array($get_num_posts_res)) {
39:             $num_posts = $posts_info['post_count'];
40:         }
41:
42:         //add to display
43:         $display_block .= <<<END_OF_TEXT
44:         <tr>
45:             <td><a href="showtopic.php?topic_id=$topic_id">
46:             <strong>$topic_title</strong></a><br/>
47:             Created on $topic_create_time by $topic_owner</td>
48:             <td class="num_posts_col">$num_posts</td>
49:         </tr>
50: END_OF_TEXT;
51:     }
52:     //free results
53:     mysqli_free_result($get_topics_res);
54:     mysqli_free_result($get_num_posts_res);
55:
56:     //close connection to MySQL
57:     mysqli_close($mysqli);
58:
59:     //close up the table
60:     $display_block .= "</table>";
61: }
62: ??
63: <!DOCTYPE html>
64: <html>
65: <head>
66: <title>Topics in My Forum</title>
67: <style type="text/css">
68:     table {
69:         border: 1px solid black;
70:         border-collapse: collapse;
71:     }
72:     th {
73:         border: 1px solid black;
74:         padding: 6px;
75:         font-weight: bold;
76:         background: #ccc;
77:     }
78:     td {
79:         border: 1px solid black;
80:         padding: 6px;
81:     }
82:     .num_posts_col { text-align: center; }
83: </style>
84: </head>
85: <body>
86: <h1>Topics in My Forum</h1>
87: <?php echo $display_block; ??
88: <p>Would you like to <a href="addtopic.html">add a topic</a>?</p>
89: </body>
90: </html>

```

---

第6行到第11行给出了第一个数据库查询，并且这个特定的查询按照日期的降序来选取所有的主题信息。换句话说，按照这样的方式收集数据：最近创建的主题出现在列表的顶部。在这个查询中，注意 `date_format()` 函数的使用，它将会创建一个比存储在数据库中的原始值更漂亮的日期显示。

第 13 行检查了查询所返回的记录的存在性。如果没有返回记录，表明表中没有主题，我们将希望告诉用户。第15行创建了一条消息。此时，如果不存在主题，脚本将会跳出 `if...else` 结构并且就此结束；下一个操作将会在第63行发生，也就是静态HTML的开始。如果脚本在这里结束，第15行中创建的消息将会在第87行显示。

然而，如果 `forum_topics` 表中有了主题，脚本将在第16行继续。在第18行，一段文本赋给了 `$display_block` 变量，其中包含了一个HTML表格的开始。第19行到第23行设置了一个具有两列的表格：一列表示标题，一列表示帖子的数目。在第26行，我们开始遍历查询的结果。

第 26 行中的 `while` 循环确保有元素从结果集中提取出来，把每一行作为一个名为 `$topic_info` 的数组提取，并且使用字段名作为数组元素来向一个新变量赋值。因此，在第27行，我们要提取的第一个元素是 `topic_id` 字段。把 `$topic_info ['topic_id']` 的值赋给 `$topic_id` 变量，意味着我们从名为 `$topic_info` 的数组中获得了 `$topic_id` 的一个局部值，该数组包含一个名为 `topic_id` 的字段。继续在第28行到第30行对 `$topic_title`、`$topic_create_time`、`$topic_owner` 变量这么做。`stripslashes()` 函数删除了在记录插入的时候输入到表中的任何转义字符。



在第33行到第36行，在while循环之中，我们执行了另一个查询来获取某个特定主题的帖子总数。在第43行，我们继续创建\$display\_block字符串，使用连接操作符(.=)来确保这个字符串已经附加到目前为止已经构建的显示字符串的后面。在第45行到第47行，我们创建了HTML表格来显示出到showtopic.php文件的链接，这个文件将显示出主题以及主题的所有者和创建时间。

在第48行，第二个HTML表格列显示了帖子的数目。在第51行，我们跳出了while循环，并且在第60行向\$display\_block字符串添加最后一部分以结束表格。剩下的代码显示了页面的HTML，包括\$display\_block字符串的值。

如果我们把这个文件保存为topiclist.php并将其放置到Web服务器文档根目录下，并且如果在数据库表中有主题，我们会看到如图21-4所示的结果。

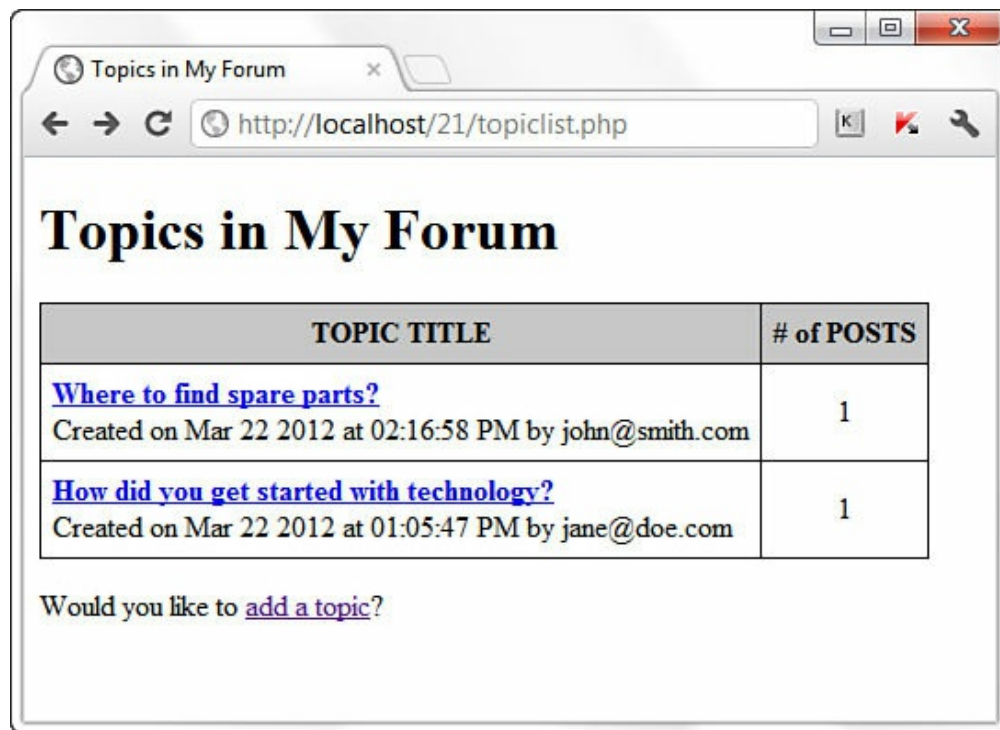


图21-4 可用主题



```

20: if (mysqli_num_rows($verify_topic_res) < 1) {
21:     //this topic does not exist
22:     $display_block = "<p><em>You have selected an invalid topic.<br/>
23:     Please <a href=\"\"topiclist.php\">try again</a>.</em></p>";
24: } else {
25:     //get the topic title
26:     while ($topic_info = mysqli_fetch_array($verify_topic_res)) {
27:         $topic_title = stripslashes($topic_info['topic_title']);
28:     }
29:
30:     //gather the posts
31:     $get_posts_sql = "SELECT post_id, post_text,
        DATE_FORMAT(post_create_time,
32:                     '%b %e %Y<br/>%r') AS fmt_post_create_time, post_owner
33:                     FROM forum_posts
34:                     WHERE topic_id = '". $safe_topic_id. "'
35:                     ORDER BY post_create_time ASC";
36:     $get_posts_res = mysqli_query($mysqli, $get_posts_sql)
37:     or die(mysqli_error($mysqli));
38:
39:     //create the display string
40:     $display_block = <<<END_OF_TEXT
41:     <p>Showing posts for the <strong>$topic_title</strong> topic:</p>
42:     <table>
43:     <tr>
44:     <th>AUTHOR</th>
45:     <th>POST</th>
46:     </tr>
47:     END_OF_TEXT;
48:
49:     while ($posts_info = mysqli_fetch_array($get_posts_res)) {
50:         $post_id = $posts_info['post_id'];
51:         $post_text = nl2br(stripslashes($posts_info['post_text']));
52:         $post_create_time = $posts_info['fmt_post_create_time'];
53:         $post_owner = stripslashes($posts_info['post_owner']);
54:
55:         //add to display
56:         $display_block .= <<<END_OF_TEXT
57:         <tr>
58:         <td>$post_owner<br/><br/>
59:         created on:<br/>$post_create_time</td>
60:         <td>$post_text<br/><br/>
61:         <a href="replytopost.php?post_id=$post_id">
62:         <strong>REPLY TO POST</strong></a></td>
63:         </tr>
64:         END_OF_TEXT;
65:     }
66:
67:     //free results
68:     mysqli_free_result($get_posts_res);
69:     mysqli_free_result($verify_topic_res);
70:
71:     //close connection to MySQL
72:     mysqli_close($mysqli);
73:
74:     //close up the table
75:     $display_block .= "</table>";
76: }

```

```
77: ?>
78: <!DOCTYPE html>
79: <html>
80: <head>
81: <title>Posts in Topic</title>
82: <style type="text/css">
83:     table {
84:         border: 1px solid black;
85:         border-collapse: collapse;
86:     }
87:     th {
88:         border: 1px solid black;
89:         padding: 6px;
90:         font-weight: bold;
91:         background: #ccc;
92:     }
93:     td {
94:         border: 1px solid black;
95:         padding: 6px;
96:         vertical-align: top;
97:     }
98:     .num_posts_col { text-align: center; }
99: </style>
100: </head>
101: <body>
102: <h1>Posts in Topic</h1>
103: <?php echo $display_block; ?>
104: </body>
105: </html>
```

---

第15行到第18行给出了这些查询中的一个，并且这个查询被用来验证在查询字符串中发送的`topic_id`实际上是一个有效的条目，它通过为所涉及的主题选择相关的`topic_title`来做到这一点。如果第20行中的验证失效，将会在第22行到第23行产生一条消息，并且脚本将跳出`if...else`语句且通过显示HTML来结束。输出如图21-5所示。

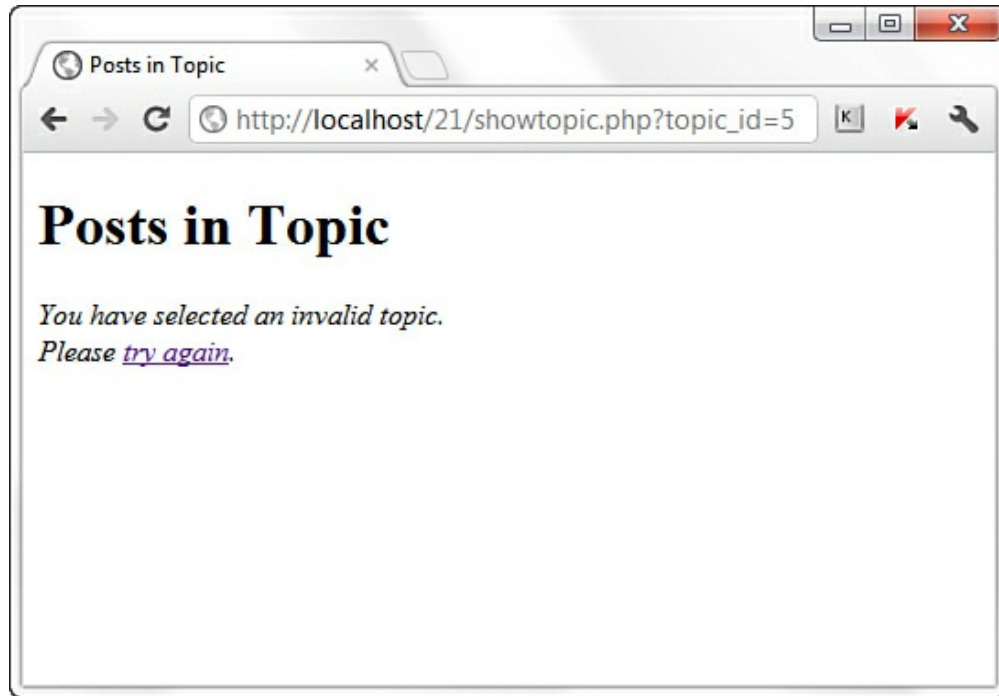


图21-5 无效的主题选取

然而，如果主题是有效的，我们在第27行提取`topic_title`的值，再次使用`stripslashes()`来删除任何转义字符。接下来，在第31行到第37行执行一个查询来按照时间的升序收集和该主题相关的所有帖子。在这个例子中，最新的帖子位于列表的底部。在第40行，开始了一段文本，其中包含了一个HTML表格的开始。第42行到第46行设置了一个具有两列的表格：其中一个用于帖子的作者，而另一个用于帖子文本本身。我们暂停编写文本块，并且在第49行开始遍历最初的查询的结果。

第49行的`while`循环确保了，虽然有从结果集提取的元素，但把每一行作为一个名为`$posts_info`的数组来提取，并且把字段名作为数组元素来向一个新的变量赋值。因此，在第50行，我们试图提取的第一个元素是`post_id`字段。我们把`$posts_info ['post_id']`的值赋给了`$post_id`变量，意味着我们从一个名为`$posts_info`的数组获取了`$post_id`的一个局部

值，这个数组包含一个名为`post_id`的字段。在第51行到第53行继续对`$post_text`、`$post_create_time`和`$post_owner`这么做。`stripslashes()`函数再次用来删除任何转义字符，并且在`$posts_info ['post_text']`的值上使用`nl2br()`函数，把所有的换行符替换为XHTML兼容的换行符。

在第56行，我们继续写入`$display_block`字符串，使用连接操作符`(.=)`来确保这个字符串添加到目前为止已经创建的字符串的末尾。在第58行到第59行，我们创建了HTML表格的一列来显示帖子的作者和创建时间。在第60行到第63行，第二个HTML表行显示了帖子的文本以及一个用来回复帖子的链接。在第65行，我们跳出了`while`循环，并且在第75行向`$display_block`字符串添加了最后一部分并且结束了这个表。其余的代码显示了这个页面的HTML，包括`$display_block`字符串的值。

如果我们把这个文件保存为`showtopic.php`并且将其放置到Web服务器文档根目录下，如果已经在数据库中有了帖子，将会看到如图21-6所示的结果。



图21-6 一个主题中的帖子

只有一个帖子的主题令人尴尬，因此，让我们通过创建向主题添加帖子的脚本来结束本章。



## 21.6 向主题添加帖子

在最后的这个步骤中，我们将创建replytopost.php脚本，其中包含了看上去和用来添加新主题的脚本类似的代码。程序清单21.6给出了这个一体化的表单和脚本，它首先在第2行和第3行开始函数文件的包含和数据库连接的初始化。尽管这个脚本根据表单的状态（表单是正在显示还是已经提交）来执行不同的任务，这两个条件下都需要在某个时刻和数据库交互。

程序清单21.6 向主题添加回复的脚本

---

```
1: <?php
2: include 'ch21_include.php';
3: doDB();
4:
5: //check to see if we're showing the form or adding the post
6: if (!$_POST) {
7:     // showing the form; check for required item in query string
8:     if (!isset($_GET['post_id'])) {
9:         header("Location: topiclist.php");
10:        exit;
11:    }
12:
13:    //create safe values for use
14:    $safe_post_id = mysqli_real_escape_string($mysqli, $_GET['post_id']);
15:
16:    //still have to verify topic and post
17:    $verify_sql = "SELECT ft.topic_id, ft.topic_title FROM forum_posts
18:                  AS fp LEFT JOIN forum_topics AS ft ON fp.topic_id =
19:                  ft.topic_id WHERE fp.post_id = '". $safe_post_id . "'";
20:
```

```

21:     $verify_res = mysqli_query($mysqli, $verify_sql)
22:         or die(mysqli_error($mysqli));
23:
24:     if (mysqli_num_rows($verify_res) < 1) {
25:         //this post or topic does not exist
26:         header("Location: topiclist.php");
27:         exit;
28:     } else {
29:         //get the topic id and title
30:         while($topic_info = mysqli_fetch_array($verify_res)) {
31:             $topic_id = $topic_info['topic_id'];
32:             $topic_title = stripslashes($topic_info['topic_title']);
33:         }
34:     ?>
35: <!DOCTYPE html>
36: <html>
37: <head>
38: <title>Post Your Reply in <?php echo $topic_title; ?></title>
39: </head>
40: <body>
41: <h1>Post Your Reply in <?php echo $topic_title; ?></h1>
42: <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
43: <p><label for="post_owner">Your Email Address:</label><br/>
44: <input type="email" id="post_owner" name="post_owner" size="40"
45: maxlength="150" required="required"></p>
46: <p><label for="post_text">Post Text:</label><br/>
47: <textarea id="post_text" name="post_text" rows="8" cols="40"
48: required="required"></textarea></p>
49: <input type="hidden" name="topic_id" value="<?php echo $topic_id; ?>">
50: <button type="submit" name="submit" value="submit">Add Post</button>
51: </form>
52: </body>
53: </html>
54: <?php
55:     }
56:     //free result
57:     mysqli_free_result($verify_res);
58:
59:     //close connection to MySQL
60:     mysqli_close($mysqli);
61:
62: } else if ($_POST) {
63:     //check for required items from form
64:     if ((!$_POST['topic_id']) || (!$_POST['post_text']) ||
65:         (!$_POST['post_owner'])) {
66:         header("Location: topiclist.php");
67:         exit;
68:     }
69:
70:     //create safe values for use
71:     $safe_topic_id = mysqli_real_escape_string($mysqli, $_POST['topic_id']);
72:     $safe_post_text = mysqli_real_escape_string($mysqli, $_POST['post_text']);
73:     $safe_post_owner = mysqli_real_escape_string($mysqli, $_POST['post_owner']);
74:
75:     //add the post
76:     $add_post_sql = "INSERT INTO forum_posts (topic_id,post_text,
77:         post_create_time,post_owner) VALUES
78:         ('.$safe_topic_id.', '$safe_post_text.',

```

```
79:             now(), '$safe_post_owner.'');";
80: $add_post_res = mysqli_query($mysqli, $add_post_sql)
81:             or die(mysqli_error($mysqli));
82:
83: //close connection to MySQL
84: mysqli_close($mysqli);
85:
86: //redirect user to topic
87: header("Location: showtopic.php?topic_id=".$_POST['topic_id']);
88: exit;
89: }
90: ?>
```

---

第6行检查表单是否已经提交。如果`$_POST`还没有值，表明表单还没有提交，并且必须显示它。然而，在显示表单之前，必须检查一个必需的项目，第8行到第11行检查GET查询字符串中的`post_id`是否有一个值存在。如果`$_GET['post_id']`中没有值存在，用户重定向到主题列表页面。

如果我们对于`$_GET['post_id']`中的值的检查通过了，第17行到第22行执行一个看似复杂的查询，它根据我们所知道的唯一的值

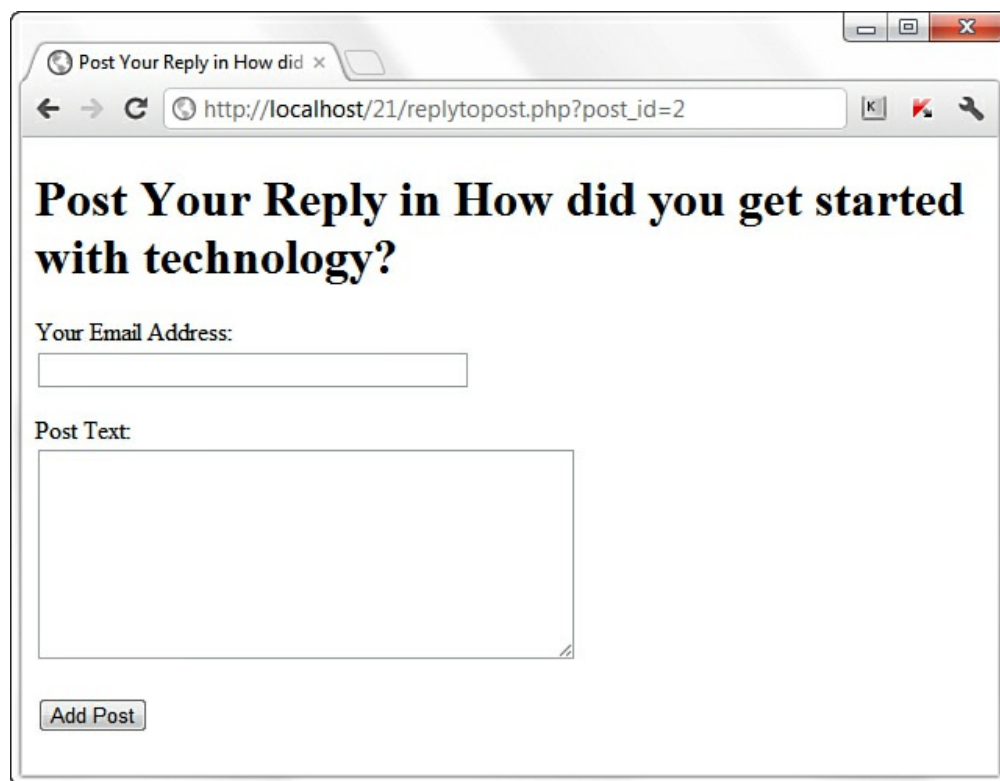
（`$_GET['post_id']`的值）从`forum_topics`表获取`topic_id`和`topic_title`字段的值。这个查询既验证了帖子的存在性，也获取了我们在后面的脚本中将要用到的信息。第24行到第27行根据这一验证测试的结果来执行，如果测试失败，就将用户重定向到`topiclist.php`页面。

如果`$_GET['post_id']`的值表示一个有效的帖子，我们在第30行到第33行提取`topic_id`和`topic_title`的值，再次使用`stripslashes()`函数来删除任何转义字符。接下来，用来添加一个帖子的表单显示在整个屏幕上，直到点击提交按钮，这个脚本就是这样。在这个表单中，我们看到动作是在第42行的`$_SERVER['PHP_SELF']`，表示这个脚本将再次调用到动作

执行中。第49行的一个隐藏字段保存了需要传递到脚本的下一次迭代中的信息。

转向第62行，当脚本重新载入并且\$\_POST包含一个值的时候，这段代码将会执行。这段代码检查了来自表单的所有必需的字段的存在（第64行到第68行），然后，如果它们都存在，使用第71行到第73行所创建的安全值，提交查询来向数据库添加帖子（第76行到第81行）。当帖子添加到数据库之后，重定向到showtopic.php页面（第87行到第88行），使用相应的查询字符串来显示活动的主题。

如果我们把这个文件保存为replytopost.php并且将其放置到Web服务器文档根目录下，自己尝试后将会看到如图21-7和图21-8所示的结果。



Post Your Reply in How did you get started with technology?

Your Email Address:

Post Text:

Add Post

图21-7 准备添加一个帖子

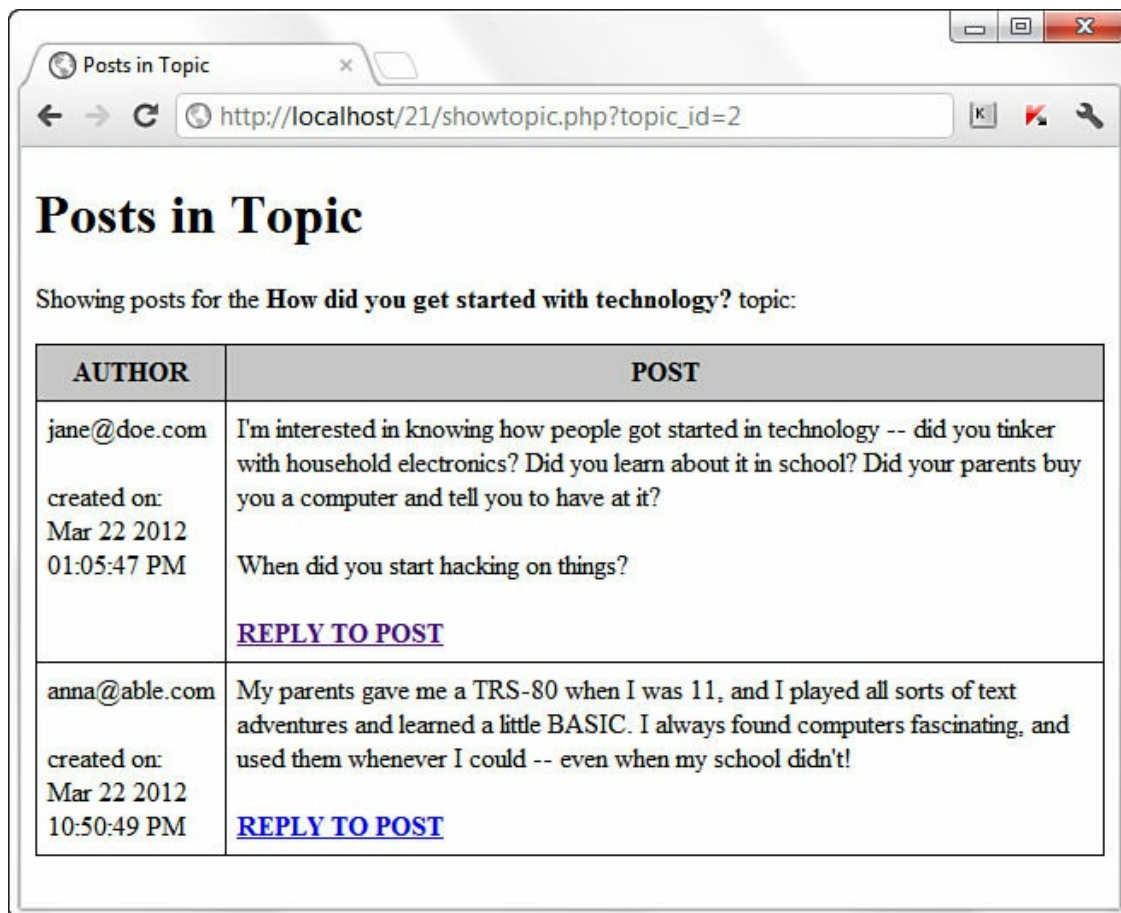


图21-8 添加到列表的一个帖子

## 21.7 小结

要采取从初学到熟练的思想，我们需要遵循一个设计过程。这个设计过程基本上可以概括为“先想后做”。讨论规则、需求和目标，然后创建规范化的表的最终版本。

在本章，我们看到了论坛为什么本质上是层级的：论坛包括主题，主题包括帖子。我们不能拥有一个没有帖子的主题，并且帖子也不会存在于不属于任何主题的论坛中。我们把这一知识应用到用来存储论坛主题和帖子的表的创建中，并且使用PHP脚本来创建这些项的输入和输出页面。

## 21.8 Q&A

**Q:** 如果我们想要有多个论坛，该怎么办？假设只有一个论坛系统可用。

**A:** 如果我们想在讨论板块上拥有多个论坛，创建一个名为forums或者具有类似效果的名字的表，它包含一个ID字段、name字段并且可能还有一个论坛说明字段。然后，在forum\_topics和forum\_posts表中，添加一个名为forum\_id的字段，以便这些在层级中较低的元素可以绑定到主论坛。确保修改用来插入记录的SQL语句，考虑到forum\_id的值。

接下来，从论坛层级开始显示，而不是从主题层级开始。就像我们不能先创建一个显示主题的脚本，然后再创建一个显示论坛的脚本。到论坛显示的链接应该包含forum\_id，并且页面自身应该在该论坛中显示所有的主题。

## 21.9 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。



## 问答题

1. 主题ID的值是如何传递到showtopic.php脚本中的？
2. 除了告诉用户主题被成功地添加了，在do\_addtopic.php脚本的末尾我们还能做什么？
3. 这段脚本为何根据表单中的值使用mysqli\_real\_escape\_string()函数。

## 解答

1. 通过超全局变量`$_GET`，即`$_GET['topic_id']`的值。
2. 就像`replytopost.php`脚本一样，我们可以删除消息显示而直接把用户重定向到她刚刚创建的主题，显示新主题及其中的所有帖子。
3. `mysqli_real_escape_string()`函数通过准备“安全的”字符串以将其插入到数据库表中，从而预防SQL注入式攻击。

## 思考题

1. 你将注意到没有页面会真正地 and 任何类型的导航绑定到一起。获取这些基本的框架脚本并且将一些导航流程用到其中。确保用户总是可以添加一个主题或者从任何给定的页面返回到主题列表。

2. 如果你感到意犹未尽，使用Q&A部分提供的信息来把多个论坛整合并显示到紧凑而小巧的讨论板块中。当你这么做的时候，应用一些文本样式和颜色，给原本简陋的示例添加一些亮色。

## 第22章 创建一个在线商店

在本章中，你将会学到：

- 为一个在线商店创建关系表。
- 创建用来显示商品分类的脚本。
- 创建显示个别商品的脚本。

在这个简短的动手实践课程中，我们将创建一个通用的在线商店。你将学会创建相关数据库表的方法，以及用来为用户显示信息的脚本。本章中使用的这个例子表示了完成这些任务的无数多种可能性方法中的一种，并且本例用来为你提供基础的知识，而不是完成这一任务的确定方法。

# 22.1 规划和创建数据库表

在我们为一个在线商店处理创建数据库的过程之前，考虑一下现实生活中的购物过程。当我们走进一家商店，商品都是按照某种顺序排列：计算机硬件和婴儿的衣服是不会放在一起的，电器和洗衣粉不会挨着摆放，等等。把这些知识应用到数据库规范化中，你就能明白需要一个表来存储分类，一个表来存储商品。在这个简单的商店中，所有商品都属于某一个类别。

接下来，考虑一下商品本身。根据我们所拥有的商店的类型，商品可能有也可能没有颜色，可能有大小也可能没有大小。但是，所有的商品都有一个名字、一个说明以及一个价格。我们再一次考虑规范化的问题，你可以看到自己拥有一个通用商品表以及和通用商品表相关的两个其他的表。

表22-1显示了在线商店的示例表和字段名。只需要一分钟的时间，就可以编写实际的SQL语句，但是，首先你需要看看这些信息并且尝试看看其关系。自己思考一下哪个字段应该是主键或唯一键。

表22-1 商店表和字段名

表 名	字 段 名
store_categories	id, cat_title, cat_desc
store_items	id, cat_id, item_title, item_price, item_desc, item_image

store_item_size	item_id, item_size
store_item_color	item_id, item_color

正如我们将在如下的SQL语句中看到的，除了id字段以外，store\_categories表还拥有两个其他的字段：cat\_title和cat\_desc，用来存储标题和说明。id字段是主键，而cat\_title是一个唯一键，因为，没有理由拥有两个相同的分类。

```
CREATE TABLE store_categories (
  id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  cat_title VARCHAR (50) UNIQUE,
  cat_desc TEXT
);
```

接下来，我们要处理store\_items表，除了id字段，它还拥有5个字段，没有一个是唯一键。字段定义中指定的长度是任意的，你应该使用最适合自己的大小。

cat\_id字段把商品和store\_categories表中的一个特定分类关联起来。这个字段不是唯一的，因为我们希望每个分类有多个商品。item\_title、item\_price和item\_desc（用于说明）字段都是一目了然的。item\_image字段将保存一个文件名，在这个例子中，假设文件是服务器的本地文件，当需要显示商品信息的时候，我们用这个文件名来构建一个HTML <img>标记。

```
CREATE TABLE store_items (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    cat_id INT NOT NULL,  
    item_title VARCHAR (75),  
    item_price FLOAT (8,2),  
    item_desc TEXT,  
    item_image VARCHAR (50)  
);
```

store\_item\_size和store\_item\_color表包含了可选的信息：如果我们销售图书，图书是不会有大小和颜色的，但如果销售衬衫，它们就有大小和颜色了。对于这些表中的每一个，都不包含键，因为我们可以把任意多种颜色和大小与一个特定商品关联起来。

```
CREATE TABLE store_item_size (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    item_id INT NOT NULL,  
    item_size VARCHAR (25)  
);  
CREATE TABLE store_item_color (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    item_id INT NOT NULL,  
    item_color VARCHAR (25)  
);
```

这就有了构建一个基本的商店所需的所有的表，也就是说，用来显示要销售的商品。第23章把用户体验集成了进来。到现在，我们只需要把注意力集中在商品目录上。

在第20章中，我们学习了如何使用PHP表单和脚本来添加或删除表中的记录。如果我们把相同的原理应用到这一组表上，可以很容易地创建一个对商店的管理前端。我们在本书中不会介绍这一过程，但你可以自己去尝试这么做。在这里，我们只是告诉你，你学习的PHP和MySQL的知识已经足够完成这个任务。

现在，我们可以通过MySQL监视器或其他界面来简单地执行

MySQL查询，向表中添加信息。如下是一些例子，如果你想要使用示例数据的话。

### 22.1.1 向store\_categories表插入记录

如下的查询在store\_categories表中创建了3个分类：hats、shirts和books。

```
INSERT INTO store_categories VALUES  
 ('1', 'Hats', 'Funky hats in all shapes and sizes!');
```

```
INSERT INTO store_categories VALUES ('2', 'Shirts', 'From t-shirts to  
sweatshirts to polo shirts and beyond.');
```

```
INSERT INTO store_categories VALUES ('3', 'Books', 'Paperback, hardback,  
books for school or play.');
```

在下一节中，我们将向分类中添加商品。

### 22.1.2 向store\_items表插入记录

如下的查询向每个分类中添加了3种商品。可以任意添加更多商品。



```

INSERT INTO store_items VALUES ('1', '1', 'Baseball Hat', '12.00',
'Fancy, low-profile baseball hat.', 'baseballhat.gif');

INSERT INTO store_items VALUES ('2', '1', 'Cowboy Hat', '52.00',
'10 gallon variety', 'cowboyhat.gif');

INSERT INTO store_items VALUES ('3', '1', 'Top Hat', '102.00',
'Good for costumes.', 'tophat.gif');

INSERT INTO store_items VALUES ('4', '2', 'Short-Sleeved T-Shirt',
'12.00', '100% cotton, pre-shrunk.', 'sstshirt.gif');

INSERT INTO store_items VALUES ('5', '2', 'Long-Sleeved T-Shirt',
'15.00', 'Just like the short-sleeved shirt, with longer sleeves.',
'lstshirt.gif');

INSERT INTO store_items VALUES ('6', '2', 'Sweatshirt', '22.00',
'Heavy and warm.', 'sweatshirt.gif');

INSERT INTO store_items VALUES ('7', '3', 'Jane\'s Self-Help Book',
'12.00', 'Jane gives advice.', 'selfhelpbook.gif');

INSERT INTO store_items VALUES ('8', '3', 'Generic Academic Book',
'35.00', 'Some required reading for school, will put you to sleep.',
'boringbook.gif');

INSERT INTO store_items VALUES ('9', '3', 'Chicago Manual of Style',
'9.99', 'Good for copywriters.', 'chicagostyle.gif');

```

#### 提示:

上面的查询引用了名称类似于“baseballhat.gif”的各种图形，而这些图形并没有包含在代码中。你可以自己找一些示例图像或自己制作一些占位符图形。

### 22.1.3 向store\_item\_size表中插入记录

下面的查询把大小和shirts分类中的3种商品中的一种关联起来，并且把一个通用的“one size fits all”（均码）的大小和hats分类中的每一种商品关联起来（假设这些帽子都是奇怪的帽子）。请你自行对shirts分类中的其他商品插入相关的同一组大小。

```
INSERT INTO store_item_size (item_id, item_size) VALUES (1, 'One Size Fits All');
INSERT INTO store_item_size (item_id, item_size) VALUES (2, 'One Size Fits All');
INSERT INTO store_item_size (item_id, item_size) VALUES (3, 'One Size Fits All');
INSERT INTO store_item_size (item_id, item_size) VALUES (4, 'S');
INSERT INTO store_item_size (item_id, item_size) VALUES (4, 'M');
INSERT INTO store_item_size (item_id, item_size) VALUES (4, 'L');
INSERT INTO store_item_size (item_id, item_size) VALUES (4, 'XL');
```

#### 22.1.4 向store\_item\_color表插入记录

如下的查询把颜色和shirts分类中的3种商品中的一种关联起来。请自行为其他的shirts和hats插入颜色记录。

```
INSERT INTO store_item_color (item_id, item_color) VALUES (1, 'red');
INSERT INTO store_item_color (item_id, item_color) VALUES (1, 'black');
INSERT INTO store_item_color (item_id, item_color) VALUES (1, 'blue');
```

## 22.2 显示商品分类

不管你是否相信，这个项目中最难的任务现在已经完成了。和考虑分类及商品相比，创建用来显示信息的脚本更加容易。我们将要创建的第一个脚本用来列出分类和商品。显然，我们不希望在用户刚一进门的时候就列出所有的分类和商品，但是确实希望给用户一个选择能够立即选取分类、查看商品，并且随后选择其他分类。换句话说，这个脚本有两个目的：它显示分类，然后如果用户点击一个分类连接，它显示该分类中的商品。

程序清单22.1给出了seestore.php的代码。如果你按照顺序阅读本书，将会注意到，在前面各章中有很多同样的基本结构；正如我在本书前面提到的，这些项目是基本的CRUD（创建、读取、更新、删除）应用程序的所有示例。即便如此，在程序清单之后，我们还是详细解释了这些代码。

程序清单22.1 浏览分类的脚本

```
1:  <?php
2:  //connect to database
3:  $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
4:
5:  $display_block = "<h1>My Categories</h1>"
6:  <p>Select a category to see its items.</p>";
7:
8:  //show categories first
9:  $get_cats_sql = "SELECT id, cat_title, cat_desc FROM
10:                  store_categories ORDER BY cat_title";
11:  $get_cats_res = mysqli_query($mysqli, $get_cats_sql)
12:                  or die(mysqli_error($mysqli));
13:
14:  if (mysqli_num_rows($get_cats_res) < 1) {
15:      $display_block = "<p><em>Sorry, no categories to browse.</em></p>";
16:  } else {
17:      while ($cats = mysqli_fetch_array($get_cats_res)) {
18:          $cat_id = $cats['id'];
19:          $cat_title = strtoupper(stripslashes($cats['cat_title']));
```

```

20:     $cat_desc = stripslashes($cats['cat_desc']);
21:
22:     $display_block .= "<p><strong><a href=\"".$_SERVER['PHP_SELF'].
23:     "?cat_id=".$cat_id.">\". $cat_title."</a></strong><br/>\"
24:     . $cat_desc."</p>";
25:
26:     if (isset($_GET['cat_id']) && ($_GET['cat_id'] == $cat_id)) {
27:         //create safe value for use
28:         $safe_cat_id = mysqli_real_escape_string($mysqli,
29:             $_GET['cat_id']);
30:
31:         //get items
32:         $get_items_sql = "SELECT id, item_title, item_price
33:             FROM store_items WHERE
34:             cat_id = '". $cat_id."' ORDER BY item_title";
35:         $get_items_res = mysqli_query($mysqli, $get_items_sql)
36:             or die(mysqli_error($mysqli));
37:
38:         if (mysqli_num_rows($get_items_res) < 1) {
39:             $display_block = "<p><em>Sorry, no items in this
40:             category.</em></p>";
41:         } else {
42:             $display_block .= "<ul>";
43:             while ($items = mysqli_fetch_array($get_items_res)) {
44:                 $item_id = $items['id'];
45:                 $item_title = stripslashes($items['item_title']);
46:                 $item_price = $items['item_price'];
47:
48:                 $display_block .= "<li><a
49:                 href=\"showitem.php?item_id=\".
50:                 $item_id.">\". $item_title."</a>
51:                 (\"$. $item_price.\")</li>";
52:             }
53:             $display_block .= "</ul>";
54:         }
55:         //free results
56:         mysqli_free_result($get_items_res);
57:     }
58: }
59: }
60: }
61: //free results
62: mysqli_free_result($get_cats_res);
63: //close connection to MySQL
64: mysqli_close($mysqli);
65: ?>
66: <!DOCTYPE html>
67: <html>
68: <head>

```

```
69: <title>My Categories</title>
70: </head>
71: <body>
72: <?php echo $display_block; ?>
73: </body>
74: </html>
```

既然我们在第20章看到过更长的代码，这个74行的功能完整的代码就比较容易接受一些。在第3行，先打开了数据库连接，因为不管脚本要采取什么动作（显示分类或者显示分类中的商品）都需要数据库。

在第5行，开始了\$display\_block字符串，将一些基本的页面标题信息添加到其中。第9行到第12行创建并执行了查询来获取分类信息。第14行检查分类，如果表中没有分类，在\$display\_block变量中存储一条消息，以便向用户显示，这就是这个脚本所做的所有的事情（它跳转到第66行的HTML，并且在空出一些数据库结果之后打印到屏幕上）。然而，如果找到分类，脚本移动到第17行，开始一个while循环来提取分类信息。

在这个while循环中，第18行到第20行获取ID、标题和分类的说明。执行了字符串操作以确保文本中没有斜杠并且分类标题是大写格式，以便显示。第22行到第24行放置分类信息，包括一个自引用的页面链接，位于\$display\_block字符串中。如果用户点击了这个链接，她将返回这个脚本，只不过将一个分类ID在查询字符串中传递。这个脚本在第26行检查这个值。

如果用户希望查看商品而点击了一个分类连接，从而使\$\_GET['cat\_id']的值传递给了脚本，这个脚本将会构建并执行另一个查询（第32行到第36行）来获取此分类中的商品。第38行到第51行检查了商品，然后构建了一个商品字符串作为\$display\_block的一部分。字符串

中的部分信息是到一个名为showitem.php的脚本的链接，这个脚本将在下一节创建。

到这里之后，脚本所要做的事情就不多了，它显示了HTML以及\$display\_block的值。图22-1显示直接访问脚本时它的输出，只有分类信息显示出来。

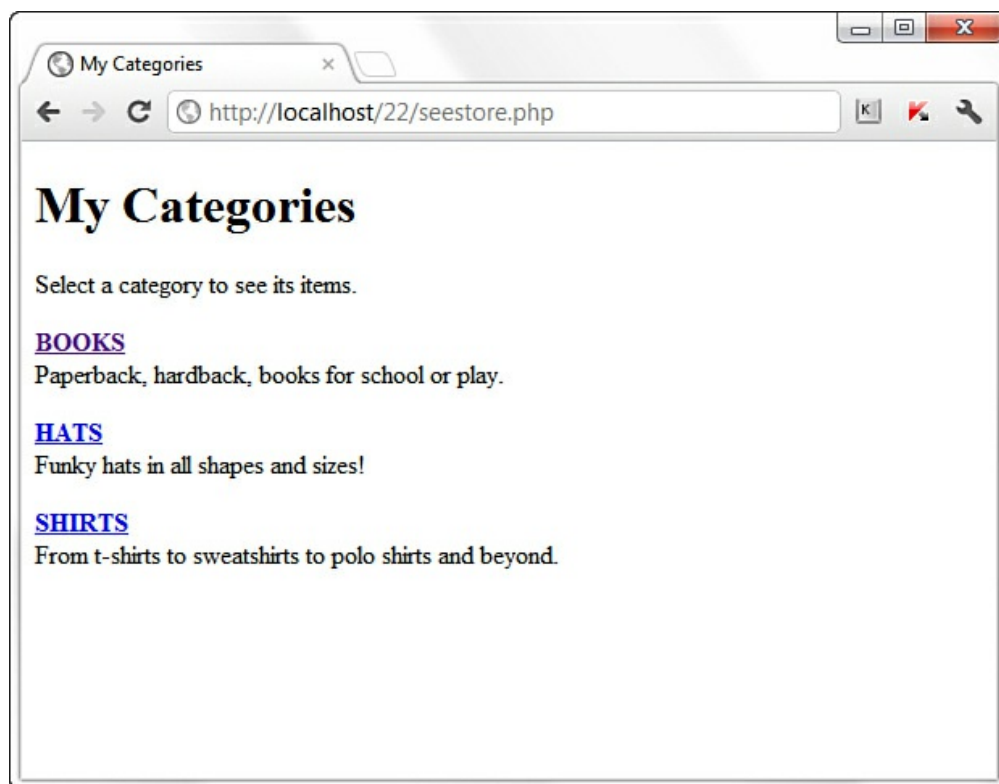


图22-1 商店中的分类

在图22-2中，我们看到了当用户点击HATS链接的时候会发生什么：脚本收集了和该分类相关的所有商品并且将它们显示在屏幕上。用户仍然可以跳转到同一个页面上的另一个分类，并且它将收集该分类的所有商品。

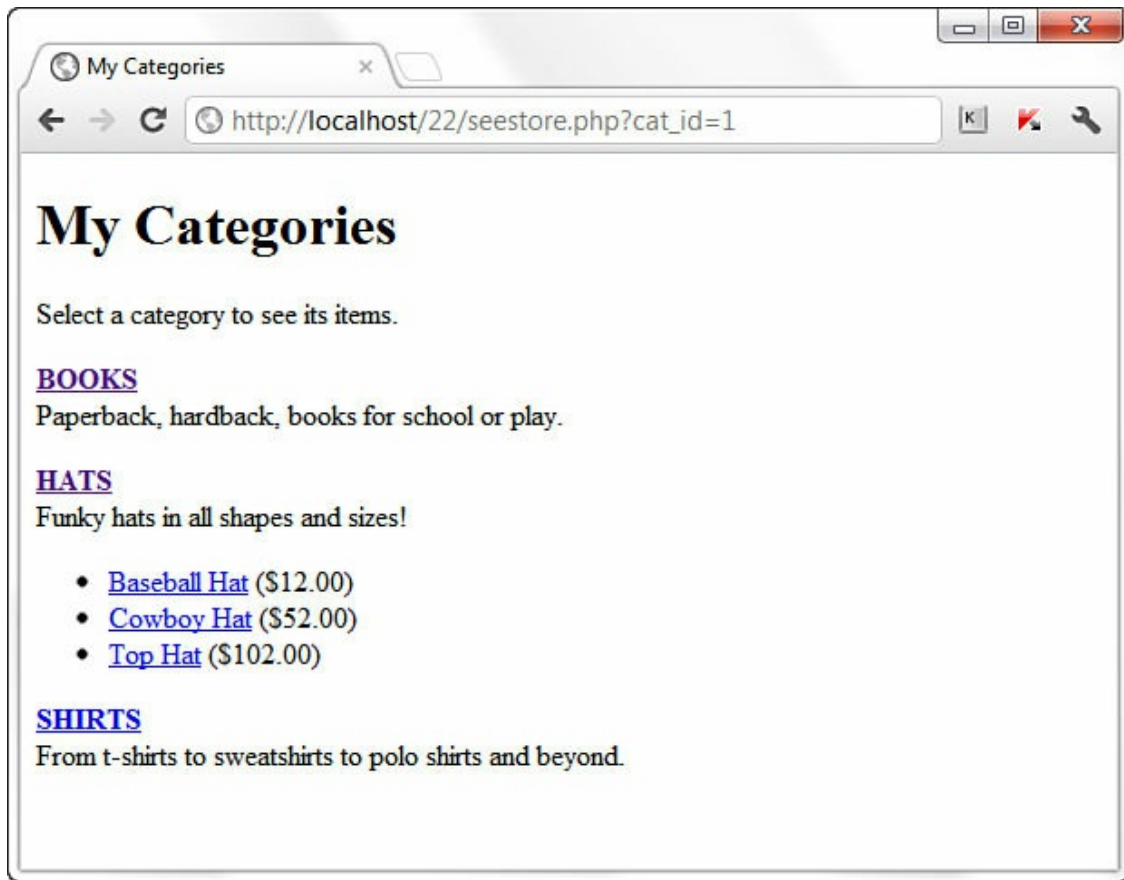


图22-2 商店的一个分类中的商品

本章的难题的最后一部分是商品显示页面的创建。



## 22.3 显示商品

本章中的商品显示页面只是显示了所有商品信息。在第23章中，我们将向其中添加一些代码，使其带有一个“add to cart”按钮的功能。程序清单22.2显示了showitem.php的代码。

程序清单22.2 浏览商品信息的脚本

```

1: <?php
2: //connect to database
3: $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
4:
5: $display_block = "<h1>My Store - Item Detail</h1>";
6:
7: //create safe values for use
8: $safe_item_id = mysqli_real_escape_string($mysqli, $_GET['item_id']);
9:
10: //validate item
11: $get_item_sql = "SELECT c.id as cat_id, c.cat_title, si.item_title,
12:                 si.item_price, si.item_desc, si.item_image FROM store_items
13:                 AS si LEFT JOIN store_categories AS c on c.id = si.cat_id
14:                 WHERE si.id = '". $safe_item_id. "'";
15: $get_item_res = mysqli_query($mysqli, $get_item_sql)
16:                 or die(mysqli_error($mysqli));
17:
18: if (mysqli_num_rows($get_item_res) < 1) {
19:     //invalid item
20:     $display_block .= "<p><em>Invalid item selection.</em></p>";
21: } else {
22:     //valid item, get info
23:     while ($item_info = mysqli_fetch_array($get_item_res)) {
24:         $cat_id = $item_info['cat_id'];
25:         $cat_title = strtoupper(stripslashes($item_info['cat_title']));
26:         $item_title = stripslashes($item_info['item_title']);
27:         $item_price = $item_info['item_price'];
28:         $item_desc = stripslashes($item_info['item_desc']);
29:         $item_image = $item_info['item_image'];
30:     }
31:
32:     //make breadcrumb trail & display of item
33:     $display_block .= <<<END_OF_TEXT
34:     <p><em>You are viewing:</em><br/>
35:     <strong><a href="seestore.php?cat_id=$cat_id">$cat_title</a> &gt;
36:         $item_title</strong></p>
37:     <div style="float: left;"></div>
39:     <div style="float: left; padding-left: 12px">
40:     <p><strong>Description:</strong><br/>$item_desc</p>
41:     <p><strong>Price:</strong> \$$item_price</p>
42:     END_OF_TEXT;
43:
44:     //free result
45:     mysqli_free_result($get_item_res);
46:

```

```

45: //get colors
46: $get_colors_sql = "SELECT item_color FROM store_item_color WHERE
47:     item_id = '". $safe_item_id.'" ORDER BY item_color";
48: $get_colors_res = mysqli_query($mysqli, $get_colors_sql)
49:     or die(mysqli_error($mysqli));
50:
51: if (mysqli_num_rows($get_colors_res) > 0) {
52:     $display_block .= "<p><strong>Available Colors:</strong><br/>";
53:     while ($colors = mysqli_fetch_array($get_colors_res)) {
54:         item_color = $colors['item_color'];
55:         $display_block .= $item_color."<br/>";
56:     }
57: }
58: //free result
59: mysqli_free_result($get_colors_res);
60:
61: //get sizes
62: $get_sizes_sql = "SELECT item_size FROM store_item_size WHERE
63:     item_id = '". $safe_item_id.'" ORDER BY item_size";
64: $get_sizes_res = mysqli_query($mysqli, $get_sizes_sql)
65:     or die(mysqli_error($mysqli));
66:
67: if (mysqli_num_rows($get_sizes_res) > 0) {
68:     $display_block .= "<p><strong>Available Sizes:</strong><br/>";
69:     while ($sizes = mysqli_fetch_array($get_sizes_res)) {
70:         $item_size = $sizes['item_size'];
71:         $display_block .= $item_size."<br/>";
72:     }
73: }
74: //free result
75: mysqli_free_result($get_sizes_res);
76:
77: $display_block .= "</div>";
78: }
79: //close connection to MySQL
80: mysqli_close($mysqli);
81: ?>
82: <!DOCTYPE html>
83: <html>
84: <head>
85: <title>My Store</title>
86: </head>
87: <body>
88: <?php echo $display_block; ?>
89: </body>
90: </html>

```

在第3行，进行了数据库的连接，因为数据库中的信息构成了这个

页面的所有内容。在第5行，开始了`$display_block`字符串，其中带有一些基本的页面标题信息。

第11行到第13行创建并执行了一个查询，来获取分类和商品信息。这个特定的查询是一个表连接。这个查询只是根据分类ID来连接表从而获取分类名称，而不是从一个表选取商品信息然后执行一个二次查询来获取分类名称。

第15行检查结果。如果表中没有匹配的商品，一条消息将显示给用户，并且脚本除此以外什么也不做。然而，如果找到了商品信息，脚本继续并且在第23行到第30行收集信息。

在第34行到第35行，我们创建了所谓的面包屑路径（`breadcrumb trail`）。这是用来回到上级页面的一个导航。上述说法的含义是“显示出一个链接以便我们能够回到分类”。这个脚本中的分类ID获取自主查询，并且附加到面包屑路径中的链接之后。

在第36行到第39行，我们继续添加`$display_block`，为有关商品的信息建立一个表格。我们使用在第23行到第30行收集的值来创建一个图像链接，显示出说明以及价格。所缺的就是颜色和大小，因此第46行到第57行选择并显示出和这一商品相关的任何颜色，并且第62行到第73行收集了和这个商品相关的尺寸。

第77行到第78结束了`$display_block`字符串和主`if...else`语句，并且由于脚本没有其他事情需要做，它显示了HTML（第82行到第90行），包括`$display_block`字符串的值。图22-3显示了当从hats分类中选择棒球帽后脚本的输出。当然，你的显示可能和我的不同，但是你应该知道其

中的原因。

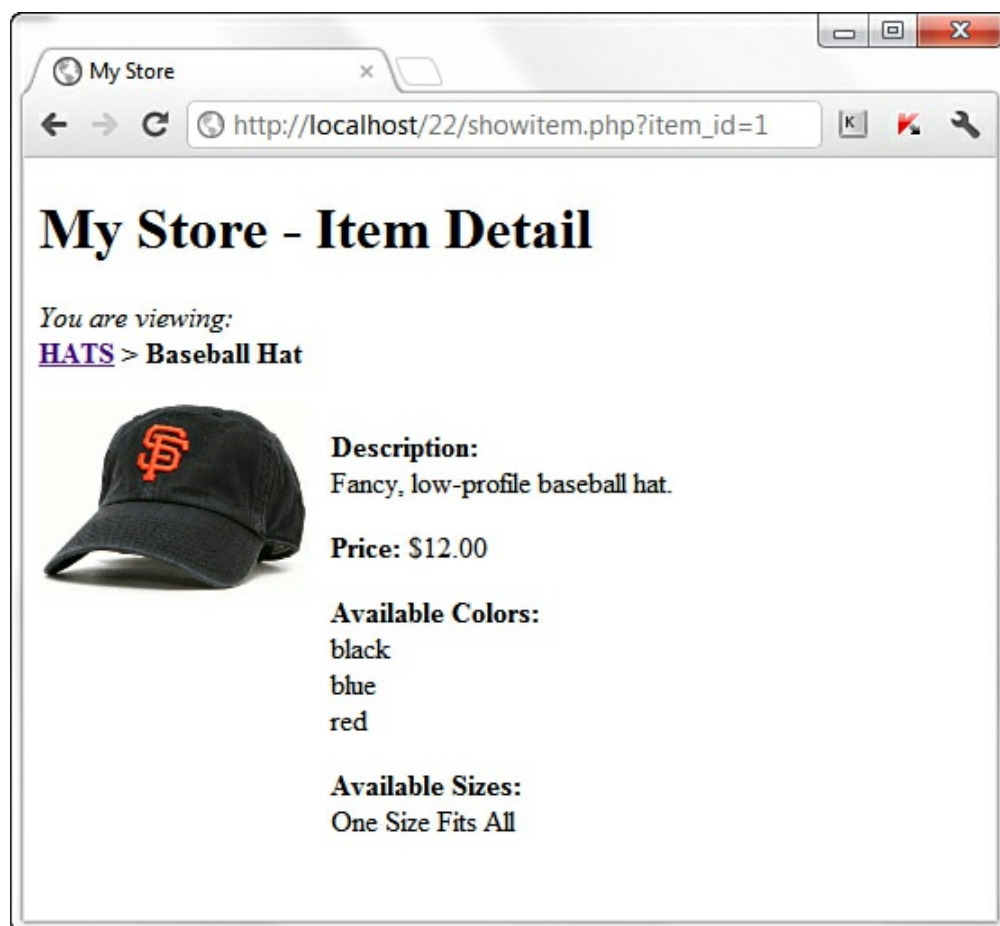


图22-3 棒球帽商品页面

这就是创建一个简单产品显示的全部内容。在第23章，我们将修改这个脚本，以便可以把商品添加到一个购物车。

## 22.4 小结

在这个动手实践的一章中，我们应用了基本的PHP和MySQL知识来创建一个商店的显示。我们学习了如何创建数据库表以及浏览分类、商品列表和单个商品的脚本。

## 22.5 Q&A

**Q:** 在商品明细记录中，我们在`item_image`字段中使用单个的文件名，如果想要链接到不同的服务器上的商品，该怎么办？

**A:** 可以在`item_image`字段输入一个URL，只要我们定义了字段来存储URL这样的长字符串。

## 22.6 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。



## 问答题

1. 哪个PHP函数用来把分类标题字符串大写？
2. 为什么store\_item\_size和store\_item\_color表没有包含任何主键或唯一键？
3. 为什么对将要用于数据库查询的值继续使用mysql\_real\_escape\_string()？

## 解答

1. `strtoupper()`

2. 根据推测，对于商品你将会具有多种颜色和多种尺寸。因此，`item_id`不是一个主键或唯一键。另外，不同商品可以具有相同的颜色或大小，因此`item_color`和`item_size`字段一定不是主键或唯一键。

3. 你应该使用`mysqli_real_escape_string()`来确保来自用户的值可以安全地使用，这些值要用于数据库的查询中，不论你是要创建一段脚本、十段脚本或一百段脚本，使用起来要足够安全。

## 思考题

1. 通过用MySQL执行自己的查询，再创建3个分类，每个分类中带有一项或两项商品。
2. 为你存储的项目中的每一项制作一些图像（或者使用创作共用许可的图像），并且，将它们放置到你自己的服务器上的一个图像目录中。对showitem.php做一些必要的修改，把该图像目录添加到生成的标签的文件路径中。

## 第23章 创建一个购物车机制

在本章中，你将学到：

- 如何为购物车和在线商店的结账部分创建一个关系表。
- 如何创建添加和删除购物车商品的脚本。
- 处理事务的一些方法，以及如何创建自己的结账流程。

在这个动手实验的一章中，我们将在第22章中已创建的商店中整合一个购物车机制和结账过程。我们将会看到创建相关数据库表的方法，以及用来添加和删除购物车商品的脚本。再一次，本章使用的例子只是展示了完成这一任务的无数种可能中的一种，并且它只是一个工作示例而不是构建在线商店的权威指南。

## 23.1 规划和创建数据库表

由于本章的目标是为用户提供选择和订购商品的一种方法，所以可以想象一下这些表是什么样的。首先，我们需要一个表来保存购物车信息。除了cart表，我们还需要一个表来保存订单，还需要一个表来保存作为每个订单的一部分的所购商品。

如下的SQL语句用来创建这3个新的表，从store\_shoppertrack表开始。当用户把商品添加到购物车的时候，这些表用来保存它们。

### 提示：

用来定义这些表的字段长度是随意选取的，为了适合多种可能的输入。请自行修改这些长度以满足你自己的特定需求。

```
CREATE TABLE store_shoppertrack (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    session_id VARCHAR (32),  
    sel_item_id INT,  
    sel_item_qty SMALLINT,  
    sel_item_size VARCHAR(25),  
    sel_item_color VARCHAR(25),  
    date_added DATETIME  
);
```

在这个表中，唯一的键就是记录的id字段。session\_id不是唯一的，否则，用户只能从商店订购一件商品，这不是做生意的好方法。

存储在session\_id字段中的值代表用户，它和分配给用户的PHP会话ID的值相匹配。sel\_\*字段保存了用户的选择：所选的商品、所选商品的数量以及所选择的商品的颜色和大小。最后，是一个date\_added字段。

很多时候，用户把商品放入到购物车中但却没有去结账。这一动作实际上在记录表中留下了散乱的商品，你可能想要定期清理。例如，你可能想删除一周之前的所有购物车中的商品，这就是date\_added字段的用武之地。

下一个表保存了订购信息。

```
CREATE TABLE store_orders (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    order_date DATETIME,  
    order_name VARCHAR (100),  
    order_address VARCHAR (255),  
    order_city VARCHAR (50),  
    order_state CHAR(2),  
    order_zip VARCHAR(10),  
    order_tel VARCHAR(25),  
    order_email VARCHAR(100),  
    item_total FLOAT(6,2),  
    shipping_total FLOAT(6,2),  
    authorization VARCHAR (50),  
    status ENUM('processed', 'pending')  
);
```

store\_orders表中的唯一的键字段是id。为了本章的简短起见，假设用户的订单和送货地址是相同的，并且这个商店只是向美国以内的地址销售商品。为送货地址信息添加另一组字段也是很容易的，如果你愿意这么做的话。如果这是一个现实的例子，你肯定想要添加字段；否则，对于那些想要购买东西作为礼物的购物者来说，你将会失去做生意的机会。

另外，这个表假设我们没有保存信用卡信息，除非能够确保对信息的超级加密或者确信自己的带有防火墙的服务器是安全的，否则不应该这么做。相反，这个表是根据实时的思路来处理信用卡。你将会在本章最后学习一些事务选项。最终保存每个订单的每行商品的表是

store\_orders\_items。

```
CREATE TABLE store_orders_items (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    order_id INT,  
    sel_item_id INT,  
    sel_item_qty SMALLINT,  
    sel_item_size VARCHAR(25),  
    sel_item_color VARCHAR(25),  
    sel_item_price FLOAT(6,2)  
);
```

sel\_\*字段看上去似曾相识，除了sel\_item\_price之外，它们和store\_shoppertrack表中的字段是相同的。主键是id字段，并且order\_id字段用来把每行商品和store\_orders中相应的记录联系起来。

这里也包含了sel\_item\_price字段，而不是简单地关联到商品记录，因为我们可能有机会要在商品记录中修改价格。如果我们在商品记录中修改价格，并且把已销售的每个商品关联到当前的新价格，我们的每个已销售商品的价格将不会反映用户实际的支付。现在，所有的表都已经解决了，下面我们将向一个用户的购物车添加商品。

## 23.2 把购物车整合到商店

在本节中，我们将修改第22章中的showitem.php脚本。目标是把商品信息页面转换为带有一个用来选择颜色、大小和数量的表单的商品信息页面。

在最初的脚本中，在第2行之前插入如下内容。

```
session_start();
```

由于购物车元素通过一个会话ID附加到用户，这个会话必须启动。下一个修改在最初脚本的第39行才出现，因此，我们的程序清单23.1从那里开始。

程序清单23.1 showitem.php中的新代码



```

39:     <p><strong>Price:</strong> \$$item_price</p>
40:     <form method="post" action="addtocart.php">
41: END_OF_TEXT;
42:
43:     //free result
44:     mysqli_free_result($get_item_res);
45:
46:     //get colors
47:     $get_colors_sql = "SELECT item_color FROM store_item_color WHERE
48:                       item_id = '". $safe_item_id.'" ORDER BY item_color";
49:     $get_colors_res = mysqli_query($mysqli, $get_colors_sql)
50:                       or die(mysqli_error($mysqli));
51:
52:     if (mysqli_num_rows($get_colors_res) > 0) {
53:         $display_block .= "<p><label for=\"sel_item_color\">
54:         Available Colors:</label><br/>
55:         <select id=\"sel_item_color\" name=\"sel_item_color\">";
56:
57:         while ($colors = mysqli_fetch_array($get_colors_res)) {
58:             $item_color = $colors['item_color'];
59:             $display_block .= "<option value=\"". $item_color."\">".
60:             $item_color."</option>";
61:         }
62:         $display_block .= "</select></p>";
63:     }
64:
65:     //free result
66:     mysqli_free_result($get_colors_res);
67:
68:     //get sizes
69:     $get_sizes_sql = "SELECT item_size FROM store_item_size WHERE
70:                      item_id = '". $safe_item_id.'" ORDER BY item_size";
71:     $get_sizes_res = mysqli_query($mysqli, $get_sizes_sql)
72:                      or die(mysqli_error($mysqli));
73:
74:     if (mysqli_num_rows($get_sizes_res) > 0) {
75:         $display_block .= "<p><label for=\"sel_item_size\">
76:         Available Sizes:</label><br/>
77:         <select id=\"sel_item_size\" name=\"sel_item_size\">";
78:
79:         while ($sizes = mysqli_fetch_array($get_sizes_res)) {
80:             $item_size = $sizes['item_size'];
81:             $display_block .= "<option value=\"". $item_size."\">".
82:             $item_size."</option>";
83:         }

```

```

84:     }
85:
86:     $display_block .= "</select></p>";
87:
88:     //free result
89:     mysqli_free_result($get_sizes_res);
90:
91:     $display_block .= "
92:     <p><label for=\"sel_item_qty\">Select Quantity:</label>
93:     <select id=\"sel_item_qty\" name=\"sel_item_qty\">";
94:
95:     for($i=1; $i<11; $i++) {
96:         $display_block .= "<option value=\"".$i.\"\">".$i."</option>";
97:     }
98:
99:     $display_block .= <<<END_OF_TEXT
100:     </select></p>
101:     <input type="hidden" name="sel_item_id"
102:     value="$_GET[item_id]" />
103:     <button type="submit" name="submit" value="submit">
104:     Add to Cart</button>
105:     </form>
106:     </div>
107: END_OF_TEXT;
108: }
109:
110: //close connection to MySQL
111: mysqli_close($mysqli);
112: ?>
113: <!DOCTYPE html>
114: <html>
115: <head>
116: <title>My Store</title>
117: <style type="text/css">
118:     label {font-weight: bold;}
119: </style>
120: </head>
121: <body>
122: <?php echo $display_block; ?>
123: </body>
124: </html>

```

第一个改变是在新代码的第40行，\$display\_block字符串追加了包含开始的<form>元素。表单的动作就是一个名为addtocart.php的脚本，我们将在下一节创建这个脚本。

下一个改变出现在第55行，`$display_block`字符串追加了包含`<select>`元素的开始标签，其名字为`sel_item_color`（“Available Colors”现在由第53行到第54行的`<label>`标签包围了起来，因为它标记一个表单元素）。在第58行到第60行，颜色放入到`<option>`元素中供用户从中选择，而不是直接显示在屏幕上。第62行结束了`<select>`元素。

对于商品大小进行了同样类似的修改。第75行到第77行，`$display_block`字符串追加了包含名为`sel_item_size`的`<select>`元素。第80行到第82行把颜色写入到`<option>`元素，并且第86行结束了`<select>`元素。

第91行到第97行是新增添到脚本中的。这些代码创建了一个名为`sel_item_qty`的`<select>`元素，用于用户选择购买多少商品。第100行结束了这个`<select>`元素，并且第101行到第102行向`item_id`添加了一个隐藏字段。第103行到第104行添加了一个提交按钮，并且第105行结束了表单。我们在第111行关闭了到MySQL的连接，剩下的代码和最初的脚本相比没什么变化。

当我们使用新版的`showitem.php`浏览棒球帽商品时，将会看到图23-1所示的结果，它反映出了表单元素的增加。

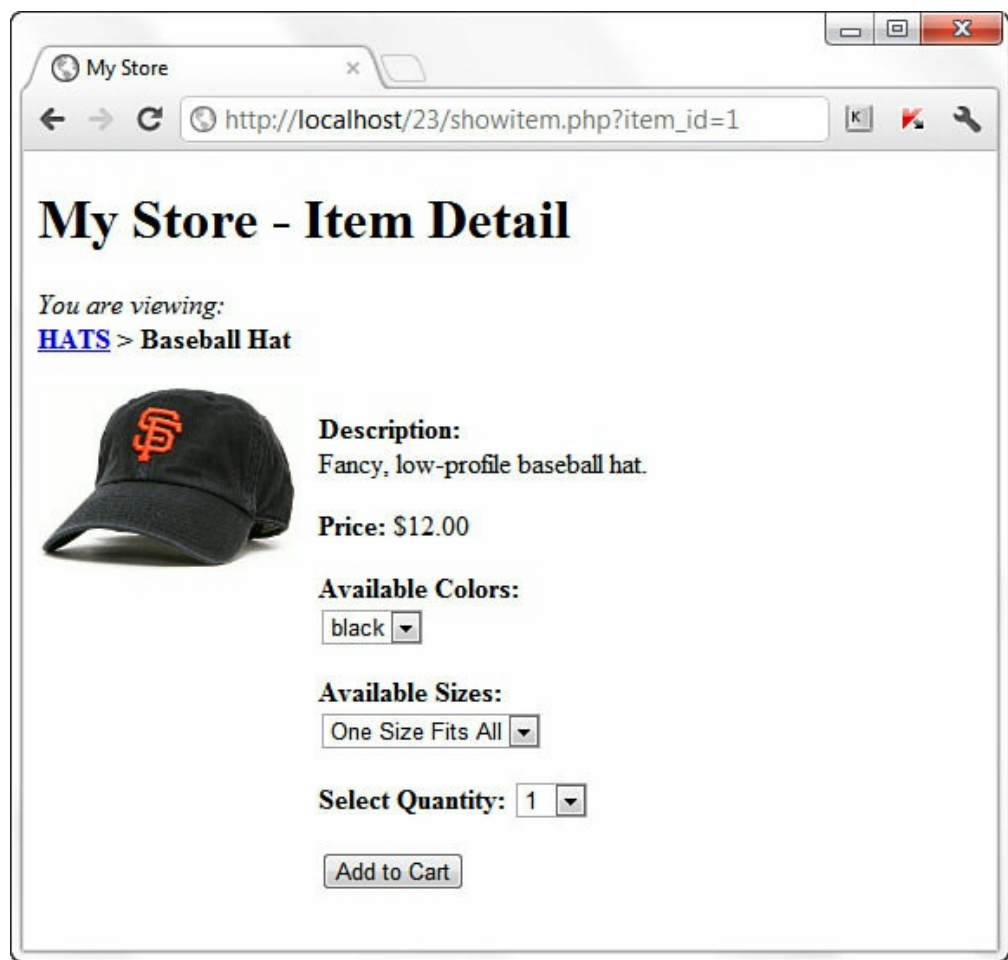


图23-1 新的棒球帽商品页面

下一步是创建addtocart.php脚本，以便刚刚创建的表单能够真正地做些事情。

### 23.2.1 把项目添加到购物车

addtocart.php脚本只是把信息写入到store\_shoppertrack表并且把用户重定向到浏览购物车。我们将首先在程序清单23.2中创建addtocart.php脚本，然后处理showcart.php脚本。

程序清单23.2 addtocart.php脚本

```

1:  <?php
2:  session_start();
3:
4:  if (isset($_POST['sel_item_id'])) {
5:      //connect to database
6:      $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
7:
8:      //create safe values for use
9:      $safe_sel_item_id = mysqli_real_escape_string($mysqli,
10:         $_POST['sel_item_id']);
11:      $safe_sel_item_qty = mysqli_real_escape_string($mysqli,
12:         $_POST['sel_item_qty']);
13:      $safe_sel_item_size = mysqli_real_escape_string($mysqli,
14:         $_POST['sel_item_size']);
15:      $safe_sel_item_color = mysqli_real_escape_string($mysqli,
16:         $_POST['sel_item_color']);
17:
18:      //validate item and get title and price
19:      $get_iteminfo_sql = "SELECT item_title FROM store_items WHERE
20:         id = '". $safe_sel_item_id. "'";
21:      $get_iteminfo_res = mysqli_query($mysqli, $get_iteminfo_sql)
22:         or die(mysqli_error($mysqli));
23:
24:      if (mysqli_num_rows($get_iteminfo_res) < 1) {
25:
26:         //free result
27:         mysqli_free_result($get_iteminfo_res);
28:
29:         //close connection to MySQL
30:         mysqli_close($mysqli);
31:
32:         //invalid id, send away
33:         header("Location: seestore.php");
34:         exit;
35:     } else {
36:         //get info
37:         while ($item_info = mysqli_fetch_array($get_iteminfo_res)) {
38:             $item_title = stripslashes($item_info['item_title']);
39:         }
40:
41:         //free result
42:         mysqli_free_result($get_iteminfo_res);
43:
44:         //add info to cart table
45:         $addtocart_sql = "INSERT INTO store_shoppertrack
46:             (session_id, sel_item_id, sel_item_qty,
47:             sel_item_size, sel_item_color, date_added)
48:             VALUES ('".$_COOKIE['PHPSESSID']."',
49:             '". $safe_sel_item_id. "',
50:             '". $safe_sel_item_qty. "',
51:             '". $safe_sel_item_size. "',
52:             '". $safe_sel_item_color. "', now())";
53:         $addtocart_res = mysqli_query($mysqli, $addtocart_sql)
54:             or die(mysqli_error($mysqli));
55:
56:

```

```
57:          //close connection to MySQL
58:          mysqli_close($mysqli);
59:
60:          //redirect to showcart page
61:          header("Location: showcart.php");
62:          exit;
63:      }
64:
65: } else {
66:     //send them somewhere else
67:     header("Location: seestore.php");
68:     exit;
69: }
70: ?>
```

第2行继续用户会话，这很重要，因为我们需要捕获用户的会话ID来写入到store\_shoppertrack表。

在第4行，这个脚本验证了出现在\$\_POST['sel\_item\_id']中的一个值，意味着当提交了相应的表单之后，用户来到这一脚本。如果没有值，脚本跳到第51行并且在第53行发送给用户，而这就是脚本所做的事情。

然而，如果\$\_POST['sel\_item\_id']中有一个值，第6行将会进行数据库连接，第9行到第16行会创建表单信息的安全版本，第17行开始将执行一个查询来验证选择的商品ID是一个有效的商品。第19行到第22行创建并执行了一个SQL查询来收集选择的商品的标题。第24行检查一个结果，如果没有结果，用户在第33行再次重定向，因为商品选择是无效的。

如果商品选择有效，脚本继续到第38行并且从结果集中提取值。这个脚本现在有足够的信息来向store\_shoppertrack表添加商品选择，这在第45行到第54行完成。

当查询执行完，用户重定向到showcart.php，其中包含所有的购物车商品。我们在下一节中创建这个脚本。

### 23.2.2 浏览购物车

既然已经向购物车添加了商品，我们肯定想要看看它们。程序清单23.3给出了showcart.php的代码。

程序清单23.3 showcart.php脚本

```
1:  <?php
2:  session_start();
3:
4:  //connect to database
```

```

5:  $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
6:
7:  $display_block = "<h1>Your Shopping Cart</h1>";
8:
9:  //check for cart items based on user session id
10: $get_cart_sql = "SELECT st.id, si.item_title, si.item_price,
11:                 st.sel_item_qty, st.sel_item_size, st.sel_item_color FROM
12:                 store_shoppertrack AS st LEFT JOIN store_items AS si ON
13:                 si.id = st.sel_item_id WHERE session_id =
14:                 '". $_COOKIE['PHPSESSID']."'";
15: $get_cart_res = mysqli_query($mysqli, $get_cart_sql)
16:                 or die(mysqli_error($mysqli));
17:
18: if (mysqli_num_rows($get_cart_res) < 1) {
19:     //print message
20:     $display_block .= "<p>You have no items in your cart.
21:     Please <a href=\"seestore.php\">continue to shop</a>!</p>";
22: } else {
23:     //get info and build cart display
24:     $display_block .= <<<END_OF_TEXT
25:     <table>
26:     <tr>
27:     <th>Title</th>
28:     <th>Size</th>
29:     <th>Color</th>
30:     <th>Price</th>
31:     <th>Qty</th>
32:     <th>Total Price</th>
33:     <th>Action</th>
34:     </tr>
35: END_OF_TEXT;
36:
37:     while ($cart_info = mysqli_fetch_array($get_cart_res)) {
38:         $id = $cart_info['id'];
39:         $item_title = stripslashes($cart_info['item_title']);
40:         $item_price = $cart_info['item_price'];
41:         $item_qty = $cart_info['sel_item_qty'];
42:         $item_color = $cart_info['sel_item_color'];
43:         $item_size = $cart_info['sel_item_size'];
44:         $total_price = sprintf("%.02f", $item_price * $item_qty);
45:
46:         $display_block .= <<<END_OF_TEXT;
47:         <tr>
48:         <td>$item_title <br></td>
49:         <td>$item_size <br></td>
50:         <td>$item_color <br></td>
51:         <td>\$ $item_price <br></td>
52:         <td>$item_qty <br></td>
53:         <td>\$ $total_price</td>
54:         <td><a href="removefromcart.php?id=$id">remove</a></td>
55:         </tr>
56: END_OF_TEXT;
57:     }
58:     $display_block .= "</table>";
59: }
60: //free result
61: mysqli_free_result($get_cart_res);
62:

```



```

63: //close connection to MySQL
64: mysqli_close($mysqli);
65: ?>
66: <!DOCTYPE html>
67: <html>
68: <head>
69: <title>My Store</title>
70: <style type="text/css">
71:     table {
72:         border: 1px solid black;
73:         border-collapse: collapse;
74:     }
75:     th {
76:         border: 1px solid black;
77:         padding: 6px;
78:         font-weight: bold;
79:         background: #ccc;
80:         text-align: center;
81:     }
82:     td {
83:         border: 1px solid black;
84:         padding: 6px;
85:         vertical-align: top;
86:         text-align: center;
87:     }
88: </style>
89: </head>
90: <body>
91: <?php echo $display_block; ?>
92: </body>
93: </html>

```

第2行继续用户会话，这很重要，因为我们需要用户会话ID来匹配store\_shoppertrack表中的记录。第5行进行数据库连接，第7行开始了\$display\_block字符串，其中带有一个页面的标题。

第10行到第14行表示一个连接查询，其中，获取了用户的保存商品。id、sel\_item\_qty、sel\_item\_size和sel\_item\_color字段都获取自store\_shoppertrack；根据匹配来自store\_shoppertrack的信息，item\_title和item\_price字段获取自store\_items表。换句话说，对于选择的商品，

Baseball Hat（棒球帽）显示为标题，而不是显示2。第15行到第16行执行了这个查询，第18行检查了结果。

如果没有结果，用户在store\_shoppertrack表中就没有商品。一条消息写入到\$display\_block字符串，脚本退出并显示消息。

如果有真正的结果，第24行到第34行创建了一个HTML表格的开头，带有为购物车中的所有（也可能是某些）信息而定义的列。第37行开始了一个while循环，从store\_shoppertrack中提取每个商品，并且这个循环持续到第56行，把信息输出到相应的表格单元格中。

在第54行，我们看到创建了一个到删除商品脚本的链接，我们将在下一节中创建这个脚本。第58行结束了这个表格，该脚本在第66行到第93行结束并且向屏幕显示HTML，包括了表格标题和其他单元格所使用的一些样式表。

现在，我们回到一个商品页面并且把商品添加到购物车。在商品写入到store\_shoppertrack表之后，我们应该重定向到showcart.php页面，并且新选择的商品也应该显示。图23-2显示了添加一些商品之后的购物车。



图23-2 添加到购物车的商品

下一步就是创建removefromcart.php脚本。

### 23.2.3 从购物车中删除项目

removefromcart.php脚本很短小，因为它所有做的就是执行一个查询并且把用户重定向到另一个脚本。不可避免的，用户想要从他的购物车中删除项目，并且这个脚本也恰恰是允许他做到这一点的。程序清单23.4显示了完整的脚本。

程序清单23.4 removefromcart.php脚本

```

1:  <?php
2:  session_start();
3:
4:  if (isset($_GET['id'])) {
5:      //connect to database
6:      $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
7:
8:      //create safe values for use
9:      $safe_id = mysqli_real_escape_string($mysqli, $_GET['id']);
10:
11:      $delete_item_sql = "DELETE FROM store_shoppertrack WHERE
12:                          id = '". $safe_id. "' and session_id =
13:                          '". $_COOKIE['PHPSESSID'] "'";
14:      $delete_item_res = mysqli_query($mysqli, $delete_item_sql)
15:                          or die(mysqli_error($mysqli));
16:
17:      //close connection to MySQL
18:      mysqli_close($mysqli);
19:
20:      //redirect to showcart page
21:      header("Location: showcart.php");
22:      exit;
23:  } else {
24:      //send them somewhere else
25:      header("Location: seestore.php");
26:      exit;
27:  }
28:  ?>

```

第2行继续用户会话，因为我们需要使用用户会话ID来和store\_shoppertrack表中的记录相匹配。第4行检查了\$\_GET['id']中的值。如果\$\_GET['id']中不存在一个值，用户不能单击来自自己的购物车的链接，从而，在第22行发送到别处。

如果\$\_GET['id']中存在一个值，第6行进行数据库连接，第9行创建该变量的一个数据库安全版本，在第14行到第15行执行第11行到第13行的那个SQL查询，并且用户在第21行重定向到showcart.php script，其中，被删除的商品将不再显示。尝试看看。

## 23.3 支付方法和结账过程

当到了为购物车中购买的商品而付款的时候，存在几种商业方法。对你来说，“正确的”方法取决于你的生意，通过银行的经商账户通常要求你有一个营业执照、一个分销许可以及其他的证书来证明你做的是合法生意。如果你只是想要卖掉一些商品的个人，可能不希望经历所有这些文书工作。然而，你仍然有选择。

不管选择什么样的支付方式，有一件事情是肯定的，如果要通过Web传递信用卡信息，必须通过一个SSL连接来实现。第30章介绍了如何获取一个SSL认证并且将其安装在你的系统上。我们不一定要在用户的整个购物过程中都使用安全连接，只是从捕获敏感信息的时候，例如结账页面，开始使用这一连接就行了。

### 23.3.1 创建结账页面

现在，你应该已经很好地掌握了一个简单页面的创建。在本章的开始，创建了带有一些字段的store\_orders表，这些字段作为页面的一个指导。

- order\_name
- order\_address
- order\_city
- order\_state
- order\_zip
- order\_tel

- order\_email

此外，我们的页面还需要用户信用卡号码、过期日期以及信用卡的名字的字段。另外一个漂亮的功能是使用一个商品小计重复了用户的购物车的内容，以便顾客能够记得他正在付款以及这笔订单大概要花多少钱。也是在结账过程的这个时刻，我们提供了可能拥有的任何送货选项。送货和销售税将在此过程的下一个步骤中计算。

从点击表单上的提交按钮开始，结账过程取决于你所使用的支付方法。下一节将介绍基本的步骤并且提供有关支付过程的各种方法的建议。

### 23.3.2 执行结账操作

如果我们已经通过银行获得了一个经商账户，可以利用像PayPal的PayFlo Pro或Authorize. Net的支付网关服务这样的实时支付服务。要了解关于这些服务的更多信息，请分别访问<http://www.authorize.net> 和 <https://www.paypal.com/webapps/mpp/merchant>。

PHP并没有包含内建的函数来支持把访问导向这些支付网关，但是，当你考虑使用这些类型的商户服务的时候，可以下载到可用于自己的应用程序的脚本，或者会得到供应用程序开发者使用的一个API的信息。

PayPal和Authorize.Net是已有的几个供商户使用的交易处理网关中的两个，我在这里介绍它们，是因为我个人从这两个网关服务诞生起就使用它们。你的银行通常会提供一个建议你使用的商户的列表。如果你不知道该从银行提供的商户列表中选择哪个，确保彻底地研究你选择的

商户，以避免资金延期并且确保你能够得到最好的服务。

在选择了一个事务处理器之后，结账脚本应该遵从如下的步骤。

1. 计算商品、税和送货费的总和。这求得一个要从信用卡扣款的总和。
2. 对总数执行信用卡授权。
3. 从卡处理程序得到一个成功的或失败的响应。如果响应失败，向用户显示一条消息，并且事务结束。如果响应成功，继续步骤4。
4. 把基本的订购信息写入到像store\_orders这样的一个表中，包括我们将接收到成功授权的授权代码。使用mysql\_insert\_id()获取这条记录的id值。
5. 对于购物车中和这个用户相关的每件商品，向store\_orders\_itemmap中插入一条记录。每条记录都将引用在上一个步骤中收集来的id（作为order\_id）。
6. 删除用户的购物车中的商品。
7. 在屏幕上显示带有授权代码而不是信用卡信息的订单，以便用户可以打印它并保存为收据。我们也可以通过Email将这一信息发送给用户。

前面列出的每个步骤，除了实际的支付授权代码，都是简单的步骤，并且我们将在本书整个过程中用到，并且，没有理由让它们变得更加复杂。

## 23.4 小结

在这个动手实验的一章中，我们应用基本的PHP和MySQL知识把一个购物车整合到第22章的商店中。包括数据库表的创建，修改商品细节页面，以及添加和删除购物车商品的新脚本。



## 23.5 Q&A

**Q:** 当用户将商品加入到自己的购物车的时候，他们如何确定该项商品有库存？

**A:** 如果修改了store\_items表，而该表包含了表示当前存货量的一个字段，并且，当用户完成了结账过程，存货量减去用户订购的书目，然后，在showitem.php脚本中，我们可以生成一个下拉列表，其中具有该商品现有库存可供选择的最大数目。当然，如果操作库存汇总的上千种商品，如果你的下拉选择器只允许每次购买10个商品的话，那没什么关系。然而，为了实现更好的用户体验，你可能需要让用户能够向购物车添加尽可能多的商品，在这种情况下，在完成添加到购物车操作之前，还需要添加一个库存检查功能，并且，不允许添加到购物车中的数量比库存中的数量还大。

## 23.6 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 当从购物车中删除一个商品时，为什么要查询根据该记录验证用户的会话ID?
2. 添加到购物车的时候，没有把价格存储在一个隐藏字段中的原因是什么？
3. 如果需要满足送货地址和付款地址不同的需求，需要对数据库和表单做些什么？

## 解答

1. 用户应该只能够删除他自己的购物车中的商品。
2. 如果我们把价格存储在一个隐藏字段中，一个恶意的用户能够在这个值发布到表单之前修改它，从而，可以在store\_shoppertrack表中写入他想要的价格，而不是实际的价格。
3. 在store\_orders表中修改已有的与地址相关的字段，以清晰地标示它们是送货地址还是付款地址，然后，在表中复制这些字段的组合（给它们一个名字，标明它们是送货地址还是付款地址），在表单和最终的INSERT语句中也这么做。

## 思考题

1. 即便你没有一个经商账户，也像23.3.1节介绍的那样，构建一个结账表单。不要忘了给这个“结账”功能添加一个到购物车（或者商店中的任何其他页面）的链接。

2. 在创建了“结账”表单之后，将订货信息和列表项保存到本章开始处创建的数据库表中。此时先不必担心经商账户或支付过程。这个过程只是，允许用户将商品保存到其购物车中，给你某个虚拟的信用卡和地址信息来结账，然后，将基本的订货信息存储到store\_orders表中，将订购的商品存储到store\_orders\_items表中。

## 第24章 创建一个简单的日历

在本章中，你将学到：

- 如何构建一个简单的日历脚本。
- 如何在日历中查看和添加事件。
- 如何构建一个类库来产生**HTML**表单中的日期下拉列表。

本章将把我们到目前为止已经学习到的有关**PHP**语言和构建应用程序的内容揉合到一起。在本章中，我们将在创建一个简单日历的背景下继续学习。

## 24.1 构建一个简单的显示日历

我们将使用在第10章中学习的日期和时间函数来构建一个日历，它将显示1980年到2010年之间的任何月份的日期（这是随机选择的年份，没有任何意义，如果你愿意，可以让自己的日历范围从1990年到2005年）。用户将能够通过下拉菜单选择年份和月份，并且选择的月份的日期按照星期几来组织排列。

在这个脚本中，我们将使用两个变量，一个用于月份，一个用于年份，这两个变量将通过用户输入提供。这些信息片断将用来根据所选月份的第一天来构建一个时间戳。如果用户输入无效或者缺失，默认值将会是当前月份的第一天。

### 24.1.1 检查用户输入

当用户第一次访问这个日历应用程序的时候，还没有提交任何信息。因此，我们必须确保脚本能够处理这样的事实，就是月和年的变量可能还没有定义。我们使用`isset()`函数来做到这一点，如果传递给它的变量还没有定义，该函数将返回`false`。然而，让我们使用`checkdate()`函数来替代它，这个函数不仅查看变量是否存在，而且还对它做一些有意义的事情，即验证它是一个日期。程序清单24.1给出了一个代码段，它检查来自一个表单的`month`和`year`变量，并根据它们构建一个时间戳。

程序清单24.1 检查来自日历脚本的用户输入

```
1: <?php
2: if ((!isset($_POST['month'])) || (!isset($_POST['year']))) {
3:     $nowArray = getdate();
4:     $month = $nowArray['mon'];
5:     $year = $nowArray['year'];
6: } else {
7:     $month = $_POST['month'];
8:     $year = $_POST['year'];
9: }
10: $start = mktime (12, 0, 0, $month, 1, $year);
11: $firstDayArray = getdate($start);
12: ?>
```

程序清单24.1只是一个较大的脚本中的一个片断，因此，它自身不会产生任何输出。但是，这是需要理解的一个重要的代码段，这也是为什么单独列出它来以便做出一些说明。

在第2行的if语句中，我们测试表单是否提供了month和year。如果month和year还没有定义，代码段中稍后所使用的mktime()函数无法从未定义的month和year参数生成一个日期。

如果这些值已经有了，我们在第3行使用getdate()来根据当前时间创建一个关联数组。随后，我们使用数组的mon和year元素来设置\$month和\$year的值（第4行和第5行）。如果已经从表单设置了这些变量，我们把数据放入到\$month和\$year变量中，免得去动最初的超全局变量\$\_POST中的值。

既然我们可以确保在\$month和\$year中有了有效的日期，就可以使用mktime()来为该月的第一天创建一个时间戳（第10行）。随后还需要有关这一时间戳的信息，因此，在第11行，我们创建了一个名为\$firstDayArray的变量，它将存储getdate()根据这一时间戳所返回的一个关联数组。



### 24.1.2 构建HTML表单

现在需要创建一个界面，以便可以让用户看到一个月份或一年的日期。为此，我们使用了SELECT元素。尽管可以用HTML直接编码，但是我们必须确保下拉列表默认的是当前选择的月份，因此，将动态地创建这一下拉列表，通过向相应的OPTION元素添加一个SELECT属性。表单在程序清单24.2中产生。

程序清单24.2 为日历脚本构建HTML表单

```

1: <?php
2: if ((!isset($_POST['month'])) || (!isset($_POST['year']))) {
3:     $nowArray = getdate();
4:     $month = $nowArray['mon'];
5:     $year = $nowArray['year'];
6: } else {
7:     $month = $_POST['month'];
8:     $year = $_POST['year'];
9: }
10: $start = mktime (12, 0, 0, $month, 1, $year);
11: $firstDayArray = getdate($start);
12: ?>
13: <!DOCTYPE html>
14: <html>
15: <head>
16: <title><?php echo "Calendar:".$firstDayArray['month']."
17: ".$firstDayArray['year']; ?></title>
18: <body>
19: <h1>Select a Month/Year Combination</h1>
20: <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
21: <select name="month">
22: <?php
23: $months = Array("January", "February", "March", "April", "May",
24: "June", "July", "August", "September", "October", "November", "December");
25: for ($x=1; $x <= count($months); $x++) {
26:     echo "<option value=\"\$x\">";
27:     if ($x == $month) {
28:         echo " selected";
29:     }
30:     echo ">".$months[$x-1]."</option>";
31: }
32: ?>
33: </select>
34: <select name="year">
35: <?php
36: for ($x=1990; $x<=2020; $x++) {
37:     echo "<option>";
38:     if ($x == $year) {
39:         echo " selected";
40:     }
41:     echo ">$x</option>";
42: }
43: ?>
44: </select>
45: <button type="submit" name="submit" value="submit">Go!</button>
46: </form>
47: </body>
48: </html>

```

已经创建了的\$start时间戳和\$firstDayArray日期出现在第2行到第11

行，让我们开始向页面写HTML。注意，在第15行和第16行，我们使用\$firstDayArray来把月份和年份添加到TITLE元素。

第20行是表单的开始。要为月份下拉列表创建SELECT元素，我们在第22行又回到了PHP模式，来编写单个的OPTION标记。首先，我们在第23行到第24行创建了一个名为\$months的数组，其中包含了12个月份的名字，以便用来显示。然后，我们遍历了这个数组，为每个名字创建了一个OPTION标记（第25行到第31行）。

如果不是我们要在第27行根据\$month变量测试\$x（for语句中的计数变量）的话，这真是编写一个简单SELECT的过于复杂的方法。如果\$x 和\$month 相等，我们把字符串SELECTED添加到OPTION标记，以确保页面载入的时候正确的月份被自动选取。在第36行到第42行，我们使用一种类似的技术来把年份写入到下拉列表中。最后，回到HTML模式，我们在第45行创建了一个提交按钮。

现在，我们有了一个表单可以向它自己发送month和year参数，并且要么默认为当前的月份和年份，要么是之前选取的年份和月份。如果我们把这个程序清单保存为dateselector.php，将其放置到Web服务器文档根目录下，然后使用Web浏览器访问它，将会看到如图24-1所示的结果（你的月份和年份可能不同）。



图24-1 日历表单

### 24.1.3 创建日历表格

现在需要创建一个表格，并且使用选定的月份的日期来填充它。我们在程序清单24.3中做到这一点，这是一个完整的日历显示脚本。

尽管第2行是新内容，但是第3行到第64行完全来自于程序清单24.2。第2行只是定义了一个常量，在这个例子中就是值为86400的ADAY (例如,“a day”)。这个值表示一天之中的秒数，我们将在这个脚本的后面用到它。

程序清单24.3 完整的日历显示脚本

```

1: <?php
2: define("ADAY", (60*60*24));
3: if ((!isset($_POST['month'])) || (!isset($_POST['year']))) {
4:     $nowArray = getdate();
5:     $month = $nowArray['mon'];
6:     $year = $nowArray['year'];
7: } else {
8:     $month = $_POST['month'];
9:     $year = $_POST['year'];
10: }
11: $start = mktime (12, 0, 0, $month, 1, $year);
12: $firstDayArray = getdate($start);
13: ?>
14: <!DOCTYPE html>
15: <html>
16: <head>
17: <title><?php echo "Calendar: ".$firstDayArray['month']."
18: ".$firstDayArray['year']; ?></title>
19: <style type="text/css">
20:     table {
21:         border: 1px solid black;
22:         border-collapse: collapse;
23:     }
24:     th {
25:         border: 1px solid black;
26:         padding: 6px;
27:         font-weight: bold;
28:         background: #ccc;
29:     }
30:     td {
31:         border: 1px solid black;
32:         padding: 6px;
33:         vertical-align: top;
34:         width: 100px;
35:     }
36: </style>
37: <body>
38: <h1>Select a Month/Year Combination</h1>
39: <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
40: <select name="month">

```

```

41: <?php
42: $months = Array("January", "February", "March", "April", "May",
43: "June", "July", "August", "September", "October", "November", "December");
44: for ($x=1; $x <= count($months); $x++) {
45:     echo "<option value=\"\$x\"";
46:     if ($x == $month) {
47:         echo " selected";
48:     }
49:     echo ">". $months[$x-1]. "</option>";
50: }
51: ?>
52: </select>
53: <select name="year">
54: <?php
55: for ($x=1980; $x<=2010; $x++) {
56:     echo "<option";
57:     if ($x == $year) {
58:         echo " selected";
59:     }
60:     echo ">$x</option>";
61: }
62: ?>
63: </select>
64: <button type="submit" name="submit" value="submit">Go!</button>
65: </form>
66: <br/>
67: <?php
68: $days = Array("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat");
69: echo "<table><tr>\n";
70: foreach ($days as $day) {
71:     echo "<td>". $day. "</td>\n";
72: }
73: for ($count=0; $count < (6*7); $count++) {
74:     $dayArray = getdate($start);
75:     if (($count % 7) == 0) {
76:         if ($dayArray['mon'] != $month) {
77:             break;
78:         } else {
79:             echo "</tr><tr>\n";
80:         }
81:     }
82:     if ($count < $firstDayArray['wday'] || $dayArray['mon'] != $month) {
83:         echo "<td>&nbsp;</td>\n";
84:     } else {
85:         echo "<td>". $dayArray['mday']. "</td>\n";
86:         $start += ADAY;
87:     }
88: }
89: echo "</tr></table>";
90: ?>
91: </body>
92: </html>

```

我们在第66行看到新代码。由于这个表格将按照星期几来索引，我们在第70行到第72行遍历星期几名称的一个数组，在第71行，将每个名称显示在其自己的表格单元格中。这个脚本的所有神奇之处在于最后，即从第73行开始的语句。

在第73行，我们初始化一个名为\$count的变量，并且确保循环在第42次迭代之后结束。这就确保了将有足够的单元格来填充日期信息，考虑到包含4个星期的一个月可能实际在月初和月末还有几天，因此，需要6个7天的周（行）。

在这个for循环中，我们使用getdate()把\$start变量转换为一个日期数组，把结果赋给\$dayArray（第73行）。尽管在循环的最初执行中，\$start是该月的第一天，对于每次迭代，我们将使用ADAY（24小时）的值来递增这个时间戳（参见第85行）。

在第75行，我们使用模除操作符根据数字7来测试\$count变量。只有当\$count是0或者是7的倍数的时候，属于这个if语句的代码块才会运行。通过这种方式，我们就知道了应该结束循环或者开始一个新的行，这里，行表示周。

当我们已经确定处于第一次迭代中或者到达了一行的末尾，可以继续在第76行执行另一个测试。如果\$dayArray的mon（月份编号）元素不再等于\$month变量，我们就结束了。别忘了，\$dayArray包含了有关\$start时间戳的信息，这就是我们所显示的月份的当前位置。当\$start超出了当前月份的范围，\$dayArray['mon']将会保存和用户输入所提供的\$month不同的一个数字。我们的模除测试显示，到达了一行的末尾，并

且我们一个新的月份意味着可以跳出循环了。然而，假设依然在所显示的月份中，我们在第79行结束一行并开始新的一行。

在第82行下一条if语句中，我们确定了是否把日期信息写入到一个单元格。并不是每个月份都从周日开始，因此，行很可能包含一个或两个空单元格。类似的，很少有月份刚好在一行末尾结束，因此，在结束表格前可能有几个空单元格。

我们已经把和一个月份的第一天相关的信息存储到`$firstDayArray`中，特别是可以访问`$firstDayArray['wday']`中表示星期几的数字了。如果`$count`的值比这个数字小，我们知道还没有到达要写入的正确的单元格。以此类推，如果`$month`变量的值不再等于`$dayArray['mon']`，我们知道到达了月末，但还没有到达行末，正如我们在前面的模除测试中所判定的那样。在这两种情况下，我们都在第83行向浏览器中写入一个空单元格。

在第67行的最后一个else子句中，我们可以做一些有趣的事情。我们已经确定在想要列出的月份之中，并且当前日期列和`$firstDayArray['wday']`中存储的星期几的数字是相等的。现在，必须使用前面在循环中建立的`$dayArray` 关联数组来把月份中的日期和一些空白写入到一个单元格中。

最后，在第69行需要递增`$start`变量，其中包含了时间戳。我们只是将其增加一天的秒数（在第2行定义了这个数字），并且，我们准备好再次测试`$start`中的新值来开始循环。如果把这个程序清单保存为`showcalendar.php`，将其放置到Web服务器的文档根目录下，并且通过Web浏览器来访问它，将会看到如图24-2所示的结果（你的月份和年份



可能有所不同）。

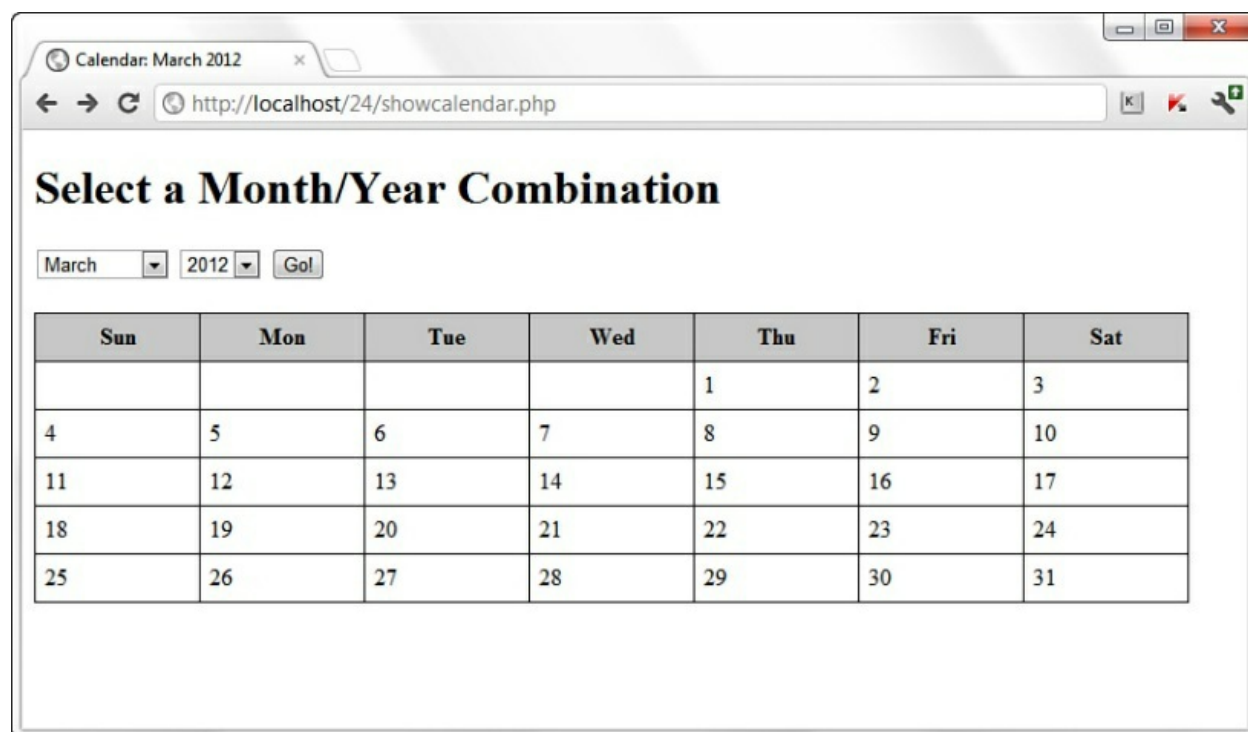


图24-2 日历表单和脚本

#### 24.1.4 向日历添加事件

显示日历是不错的事情，但是，只需要略多几行的代码，就可以让日历更具有交互性，也就是说，我们可以在给定的一天添加和浏览事件。要开始做到这一点，让我们创建一个简单的数据库表，其中存储了事件信息。为了简单起见，这些事件只在单独的一天发生，并且将只显示其开始日期和时间。尽管我们可以让事件记录尽可能地复杂，这里的这个例子只是为了展示所涉及的基本过程。

`calendar_events`表将包含用于开始日期和时间、事件标题以及事件的简短说明的字段。

```
CREATE TABLE calendar_events (
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    event_title VARCHAR (25),
    event_shortdesc VARCHAR (255),
    event_start DATETIME
);
```

我们可以使用程序清单24.3中的代码作为基础。在新的脚本中，将添加一个到弹出窗口的链接作为日历显示的一部分。每个日期都是一个链接，弹出窗口将调用另外一个脚本，该脚本显示一个事件的完整文本并且能够添加事件。为了开始做到这一点，先在最初的脚本的第36行的开始<head>标记的后面，添加如下的JavaScript代码。

```
<script type="text/javascript">
function eventWindow(url) {
    event_popupWin = window.open(url, 'event', 'resizable=yes, scrollbars=yes,
        toolbar=no,width=400,height=400');
    event_popupWin.opener = self;
}
</script>
```

这个JavaScript函数定义了一个400×400的窗口，它将调用我们所提供的一个URL。在最初脚本的第85行使用这个JavaScript函数，我们现在把这个链接中显示的日期包装到基于JavaScript的弹出窗口中，这个窗口将调用一个名为event.php的脚本。新的代码行如下所示。

```
echo "<td><a href=\"javascript:eventWindow('event.php?m=\".$month.
    \"&amp;d=\".$dayArray['mday'].\"&amp;y=$year')\">\".$dayArray['mday'].\"</a>
<br/><br/>\".$event_title.\"</td>\n\";
```

我们不仅调用event.php文件，而且必须与它一起发送被点击的特定链接的日期信息。这通过查询字符串来做到，并且我们可以看到一起发送3个变量，这就是针对月份的\$\_GET['m']，针对日期的\$\_GET['d']，和针对年份的\$\_GET['y']。

在我们处理event.php脚本之前，对于这个特定的脚本只保留了一点

更改：如果事件确实存在，我们对这个特定的视图添加一个指示器。这个查询检查某个给定日期的已有事件，它出现在最初脚本第84行的else语句的开始。出现了一个全新的else语句，我们可以看到查询在这里执行了，并且，如果找到结果，文本将在那一天的表格单元格中显示出来。

```
} else {
    $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
    $chkEvent_sql = "SELECT event_title FROM calendar_events WHERE
                    month(event_start) = '". $month. "' AND
                    dayofmonth(event_start) = '". $dataArray['mday']. "'
                    AND year(event_start) = '". $year. "' ORDER BY event_start";
    $chkEvent_res = mysqli_query($mysqli, $chkEvent_sql)
                    or die(mysqli_error($mysqli));

    if (mysqli_num_rows($chkEvent_res) > 0) {
        while ($sev = mysqli_fetch_array($chkEvent_res)) {
            $event_title = stripslashes($sev['event_title']);
        }
    } else {
        $event_title = "";
    }

    echo "<td><a href=\"javascript:eventWindow('event.php?m='". $month.
    "&d='". $dataArray['mday']. "&y=$year')\">".
    $dataArray['mday']. "</a><br/><br/>". $event_title. "</td>\n";

    unset($event_title);

    $start += ADAY;
}
```

在程序清单24.4中，我们可以看到完整的新的脚本，名为showcalendar\_withevent.php。

程序清单24.4 带有条目相关的修改的日历显示脚本

```

1:  <?php
2:  define("ADAY", (60*60*24));
3:  if ((!isset($_POST['month'])) || (!isset($_POST['year']))) {
4:      $nowArray = getdate();
5:      $month = $nowArray['mon'];
6:      $year = $nowArray['year'];
7:  } else {
8:      $month = $_POST['month'];
9:      $year = $_POST['year'];
10: }
11: $start = mktime (12, 0, 0, $month, 1, $year);
12: $firstDayArray = getdate($start);
13: ?>
14: <!DOCTYPE html>
15: <html>
16: <head>
17: <title><?php echo "Calendar: ".$firstDayArray['month']." ".
18: $firstDayArray['year'] ?></title>
19: <head>
20: <style type="text/css">
21:     table {
22:         border: 1px solid black;
23:         border-collapse: collapse;
24:     }
25:     th {
26:         border: 1px solid black;
27:         padding: 6px;
28:         font-weight: bold;
29:         background: #ccc;
30:     }
31:     td {

```

```

32:         border: 1px solid black;
33:         padding: 6px;
34:         vertical-align: top;
35:         width: 100px;
36:     }
37: </style>
38: <script type="text/javascript">
39: function eventWindow(url) {
40:     event_popupWin = window.open(url, 'event',
41:     'resizable=yes,scrollbars=yes,toolbar=no,width=400,height=400');
42:     event_popupWin.opener = self;
43: }
44: </script>
45: <body>
46: <h1>Select a Month/Year Combination</h1>
47: <form method="post" action="<?php echo $_SERVER['PHP_SELF']; ?>">
48: <select name="month">
49: <?php
50: $months = Array("January", "February", "March", "April", "May", "June",
51: "July", "August", "September", "October", "November", "December");
52: for ($x=1; $x <= count($months); $x++) {
53:     echo "<option value=\"\$x\"";
54:     if ($x == $month) {
55:         echo " selected";
56:     }
57:     echo ">". $months[$x-1]. "</option>";
58: }
59: ?>
60: </select>
61: <select name="year">
62: <?php
63: for ($x=1990; $x<=2020; $x++) {
64:     echo "<option";
65:     if ($x == $year) {
66:         echo " selected";
67:     }
68:     echo ">$x</option>";
69: }
70: ?>
71: </select>
72: <button type="submit" name="submit" value="submit">Go!</button>
73: </form>
74: <br/>
75: <?php
76: $days = Array("Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat");
77: echo "<table><tr>\n";
78: foreach ($days as $day) {
79:     echo "<th>$day</th>\n";
80: }
81:
82: for ($count=0; $count < (6*7); $count++) {
83:     $dayArray = getdate($start);
84:     if (($count % 7) == 0) {
85:         if ($dayArray['mon'] != $month) {
86:             break;
87:         } else {
88:             echo "</tr><tr>\n";
89:         }

```

```

90:     }
91:     if ($count < $firstDayArray['wday'] || $dayArray['mon'] != $month) {
92:         echo "<td>&nbsp;</td>\n";
93:     } else {
94:         $mysqli = mysqli_connect("localhost", "joeuser", "somepass",
95:             "testDB");
96:         $chkEvent_sql = "SELECT event_title FROM calendar_events WHERE
97:             month(event_start) = '". $month. "' AND
98:             dayofmonth(event_start) = '". $dayArray['mday']. "'
99:             AND year(event_start) = '". $year. "' ORDER BY
100:                 event_start";
101:         $chkEvent_res = mysqli_query($mysqli, $chkEvent_sql)
102:             or die(mysqli_error($mysqli));
103:         if (mysqli_num_rows($chkEvent_res) > 0) {
104:             while ($sev = mysqli_fetch_array($chkEvent_res)) {
105:                 $event_title .= stripslashes($sev['event_title'])."<br/>";
106:             }
107:         } else {
108:             $event_title = "";
109:         }
110:         echo "<td><a href=\"javascript:eventWindow('event.php?m='". $month.
111:             "&d='". $dayArray['mday']. "&y=$year');\">".
112:             $dayArray['mday']. "</a><br/><br/>". $event_title. "</td>\n";
113:
114:         unset($event_title);
115:         $start += ADAY;
116:     }
117: }
118: echo "</tr></table>";
119:
120: //close connection to MySQL
121: mysqli_close($mysqli);
122: ?>
123: </body>
124: </html>

```

在图24-3中，我们可以看到新的日历，包括在填写了一个事件的日期上显示事件标题，为了便于说明，在这里，我们在calendar\_events表中增加了一个事件。

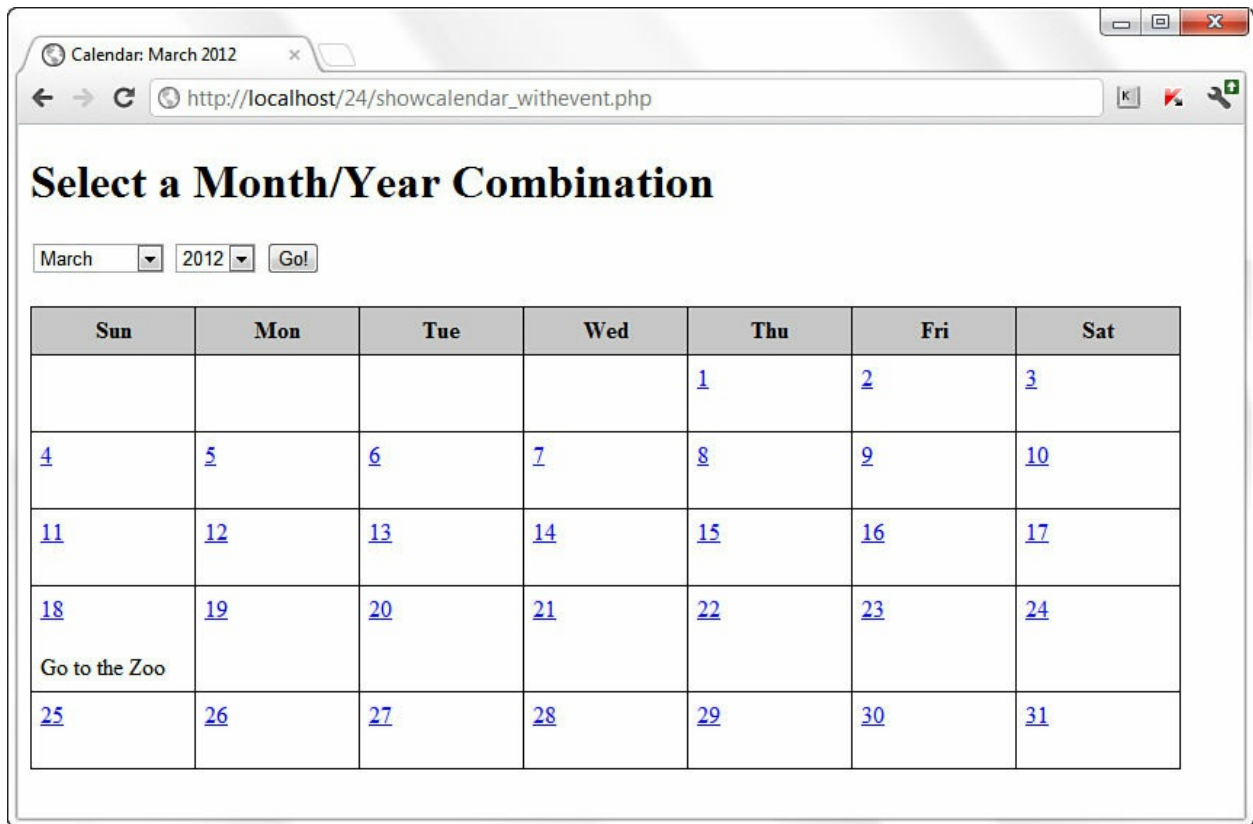


图24-3 显示带有事件的日历

剩下的就只是添加一体化的event.php脚本，这个脚本用于弹出窗口的显示并且也添加一个事件到日历（在一个特定的日期）。程序清单24.5包含了所有所需的代码，有趣的部分从第8行开始，那里连接到MySQL数据库。第11行查看事件条目表单是否已经提交，如果已经提交，在第14行到第24行创建了对数据库安全的值，创建并执行一条INSERT语句，向calendar\_events表添加事件，然后再继续执行（第29行到第34行）。

程序清单24.5 通过弹出窗口显示事件/添加事件

```

1:  <!DOCTYPE html>
2:  <html>
3:  <head>
4:  <title>Show/Add Events</title>
5:  <body>
6:  <h1>Show/Add Events</h1>
7:  <?php
8:  $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB");
9:
10: //add any new event
11: if ($_POST) {
12:
13:     //create database-safe strings
14:     $safe_m = mysqli_real_escape_string($mysqli, $_POST['m']);
15:     $safe_d = mysqli_real_escape_string($mysqli, $_POST['d']);
16:     $safe_y = mysqli_real_escape_string($mysqli, $_POST['y']);
17:     $safe_event_title = mysqli_real_escape_string($mysqli,
18:         $_POST['event_title']);
19:     $safe_event_shortcode = mysqli_real_escape_string($mysqli,
20:         $_POST['event_shortcode']);
21:     $safe_event_time_hh = mysqli_real_escape_string($mysqli,
22:         $_POST['event_time_hh']);
23:     $safe_event_time_mm = mysqli_real_escape_string($mysqli,
24:         $_POST['event_time_mm']);
25:
26:     $event_date = $safe_y."-".$safe_m."-".$safe_d."
27:         ".$safe_event_time_hh.":".$safe_event_time_mm.":00";
28:
29:     $insEvent_sql = "INSERT INTO calendar_events (event_title,
30:         event_shortcode, event_start) VALUES
31:         ('".$safe_event_title."', '".$safe_event_shortcode."',
32:         '".$event_date."')";
33:     $insEvent_res = mysqli_query($mysqli, $insEvent_sql)

```



```

34:         or die(mysql_error($mysqli));
35:
36:     } else {
37:
38:         //create database-safe strings
39:         $safe_m = mysqli_real_escape_string($mysqli, $_GET['m']);
40:         $safe_d = mysqli_real_escape_string($mysqli, $_GET['d']);
41:         $safe_y = mysqli_real_escape_string($mysqli, $_GET['y']);
42:     }
43:
44:     //show events for this day
45:     $getEvent_sql = "SELECT event_title, event_shortdesc,
46:                     date_format(event_start, '%l:%i %p') as fmt_date
47:                     FROM calendar_events WHERE month(event_start) =
48:                     '". $safe_m. "' AND dayofmonth(event_start) =
49:                     '". $safe_d. "' AND year(event_start) =
50:                     '". $safe_y. "' ORDER BY event_start";
51:     $getEvent_res = mysqli_query($mysqli, $getEvent_sql)
52:         or die(mysql_error($mysqli));
53:
54:     if (mysqli_num_rows($getEvent_res) > 0) {
55:         $event_txt = "<ul>";
56:         while ($sev = @mysqli_fetch_array($getEvent_res)) {
57:             $event_title = stripslashes($sev['event_title']);
58:             $event_shortdesc = stripslashes($sev['event_shortdesc']);
59:             $fmt_date = $sev['fmt_date'];
60:             $event_txt .= "<li><strong>". $fmt_date. "</strong>:
61:                         ". $event_title. "<br/>". $event_shortdesc. "</li>";
62:         }
63:         $event_txt .= "</ul>";
64:         mysqli_free_result($getEvent_res);
65:     } else {
66:         $event_txt = "";
67:     }
68:     // close connection to MySQL
69:     mysqli_close($mysqli);
70:
71:     if ($event_txt != "") {
72:         echo "<p><strong>Today's Events:</strong></p>
73:         $event_txt
74:         <hr/>";
75:     }
76:
77:     // show form for adding an event
78:     echo <<<END_OF_TEXT
79:     <form method="post" action="$_SERVER[PHP_SELF]">
80:     <p><strong>Would you like to add an event?</strong><br/>
81:     Complete the form below and press the submit button to
82:     add the event and refresh this window.</p>
83:

```

```

84: <p><label for="event_title">Event Title:</label><br/>
85: <input type="text" id="event_title" name="event_title"
86:     size="25" maxlength="25" /></p>
87:
88: <p><label for="event_shortdesc">Event Description:</label><br/>
89: <input type="text" id="event_shortdesc" name="event_shortdesc"
90:     size="25" maxlength="255" /></p>
91: <fieldset>
92: <legend>Event Time (hh:mm):</legend>
93: <select name="event_time_hh">
94: END_OF_TEXT;
95:
96: for ($x=1; $x <= 24; $x++) {
97:     echo "<option value=\"\$x\">$x</option>";
98: }
99:
100: echo <<<END_OF_TEXT
101: </select> :
102: <select name="event_time_mm">
103: <option value="00">00</option>
104: <option value="15">15</option>
105: <option value="30">30</option>
106: <option value="45">45</option>
107: </select>
108: </fieldset>
109: <input type="hidden" name="m" value="$safe_m">
110: <input type="hidden" name="d" value="$safe_d">
111: <input type="hidden" name="y" value="$safe_y">
112:
113: <button type="submit" name="submit" value="submit">Add Event</button>
114: </form>
115: END_OF_TEXT;
116: ?>
117: </body>
118: </html>

```

第45行到第52行执行了该查询并且获取了和给定的一天的事件对应的所有记录。用来显示条目的文本块在第54行到第67行创建。然而，用户也需要看到添加一个事件的表单，并且这个表单在第79行到第114行创建，实际上也就是这个脚本的末尾。

在图24-4中，我们看到了当从日历打开一个链接的时候，一个弹出窗口是什么样的，并且显示了一个条目。在这个例子中，我们想要在这一天添加另外一个事件，因此，这个表单已经完成并准备好添加另一个事件。

Show/Add Events - Google Chrome

localhost/24/event.php?m=3&d=1&y=2012

## Show/Add Events

**Today's Events:**

- **1:00 PM:** Go to the Zoo  
Time to see the Red Pandas!

---

**Would you like to add an event?**  
Complete the form below and press the submit button to add the event and refresh this window.

Event Title:

Event Description:

Event Time (hh:mm):  
 :

图24-4 显示了这一天的细节，准备添加另一个事件。

在图24-5中，在这个特定的日期添加了第二个事件。

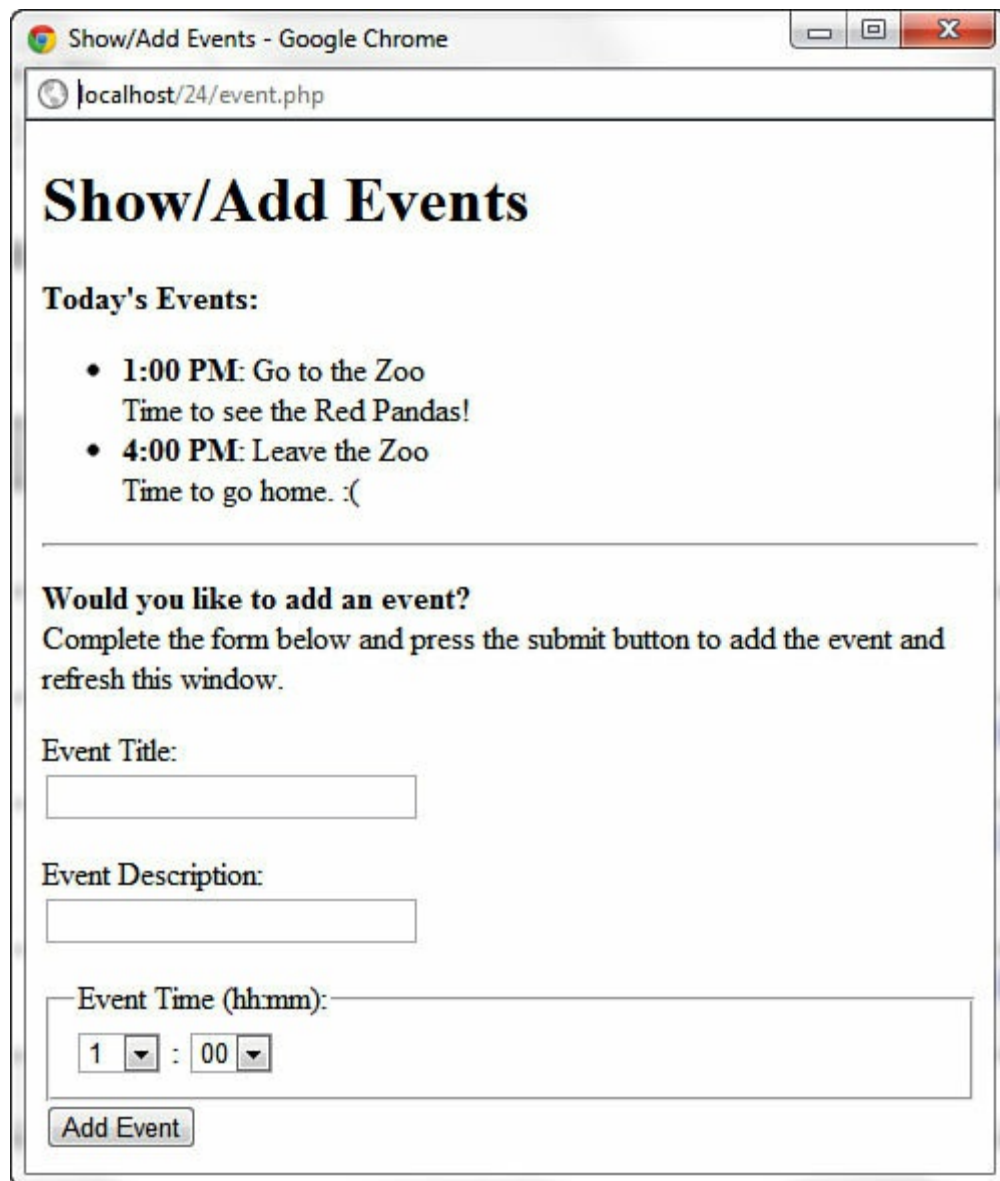


图24-5 已经添加了第二个事件

显然，这是一个简单的例子，但是它展示了构建一个日历类型的系统实际上很容易，只需要一个简短的脚本。

## 24.2 创建一个日历库

由于日期在Web界面上是随处可见的，并且由于使用日期往往相对比较复杂，让我们看看如何创建一个类库来自动化某些日期显示的工作。在这个过程中，我们将会回顾某些已经介绍过的技术，尤其是在第9章中介绍过的内容。

这个实例中创建的简单的date\_pulldown库将由3个分开的select元素组成，一个用于月份中的日期，一个用于月份，一个用于年份。

当用户提交一个页面，这个脚本将验证表单输入。如果输入没有问题，我们将使用仍然存在的用户输入来刷新这个页面。使用文本框，这很容易完成，但使用下拉菜单就有点添乱。显示从一个数据库提取的信息的页面也面临着类似的问题。日期可以直接输入到文本类型的输入元素的值属性中。日期将需要划分为月份、日期和年份值，以及选中的正确的选择元素。

date\_pulldown类的目的就是使得日期下拉菜单稳定（这样，它们可以在页面和页面之间保存设置）并且容易设置。为了创建我们的类，首先需要声明它并且创建一个构造方法。

### 提示：

构造方法存在于类中，并且当创建类的一个新的实例的时候自动调用该函数。

我们也可以声明一些类属性。我们将浏览程序清单24.6，这是显示

类的开始的第一个代码段。

程序清单24.6 创建一个日历库

```
1: class date_pulldown {
2:     public $name;
3:     public $timestamp = -1;
4:     public $months = array("Jan", "Feb", "Mar", "Apr", "May", "Jun",
5:         "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
6:     public $yearstart = -1;
7:     public $yearend = -1;
8:
9:     function date_pulldown($name) {
10:         $this->name = $name;
11:     }
```

首先在第2行声明了\$name属性，这将用来命名HTML select元素。第3行定义的\$timestamp属性，将存储一个UNIX时间戳。第4行到第5行定义的\$months数组属性，包含了我们在月份下拉列表中显示的字符串。\$yearstart和\$yearend属性（第6行和第7行）初始都设置为-1，它们最终将保存年份下拉列表中显示的第一年和最后一年的范围。

构造方法是简单的。它接受一个字符串，即我们赋给\$name属性的字符串。既然我们已经有了类的基础，还需要一组方法，以便客户代码通过它们能够设置日期。程序清单24.6从下面继续。

程序清单24.6 创建一个日历库（续）

```

12:     function setDate_global( ) {
13:         if (!$this->setDate_array($GLOBALS[$this->name])) {
14:             return $this->setDate_timestamp(time());
15:         }
16:         return true;
17:     }
18:
19:     function setDate_timestamp($time) {
20:         $this->timestamp = $time;
21:         return true;
22:     }
23:
24:     function setDate_array($inputdate) {
25:         if (is_array($inputdate) &&
26:             isset($inputdate['mon']) &&
27:             isset($inputdate['mday']) &&
28:             isset($inputdate['year'])) {
29:
30:             $this->timestamp = mktime(11, 59, 59,
31:                                     $inputdate['mon'], $inputdate['mday'], $inputdate['year']);
32:             return true;
33:         }
34:         return false;
35:     }

```

在这里所给出的方法中，`setDate_timestamp()`是最简单的（第19行到第22行）。它需要一个UNIX时间戳，这个时间戳会赋给`$timestamp`属性。但是，我们也不要忘记了其他的方法。

`setDate_array()`方法（第24行到第35行）期待一个关联数组，它至少带有3个键：`mon`、`mday`和`year`。这些键所拥有的数据值将会和`getdate()`所返回的数组中的值具有相同的格式。这意味着`setDate_array()`将会接受一个手动构建的数组，示例如下。

```
array("mday"=> 25, "mon"=>3, "year"=> 2012);
```

或者是调用`getdate()`的结果，如下所示。

```
getdate(1208052013);
```

这不是偶然，我们稍后要构建的下拉列表将会产生一个包含了

mon、mday和year键的数组。这个方法使用mktime()函数来构建一个时间戳，随后，这个时间戳将赋给\$timestamp变量。

setDate\_global()方法（第12行到第17行）默认被调用。它尝试找到和对象的\$name属性具有相同名字的一个全局变量，这个变量传递给了setDate\_array()。如果这个方法找到一个具有正确构造的全局变量，它使用这个变量来创建\$timestamp变量。否则，使用当前日期。

日期和月份的范围是固定的，但是，年份则不同了。随着程序清单24.6的继续，我们创建了一些方法，允许客户代码设置自己的年份范围，尽管我们也提供了默认的行为。

程序清单24.6 创建一个日历库(续)

```
36:     function setYearStart($year) {
37:         $this->yearstart = $year;
38:     }
39:
40:     function setYearEnd($year) {
41:         $this->yearend = $year;
42:     }
43:
44:     function getYearStart() {
45:         if ($this->yearstart < 0) {
46:             $nowarray = getdate(time());
47:             $this->yearstart = $nowarray['year']-5;
48:         }
49:
50:         return $this->yearstart;
51:     }
52:
53:     function getYearEnd() {
54:         if ($this->yearend < 0) {
55:             $nowarray = getdate(time());
56:             $this->yearend = $nowarray['year']+5;
57:         }
58:         return $this->yearend;
59:     }
```



`setYearStart()`和`setYearEnd()`方法都相当简单（第36行到第43行），因为一个年份直接地赋给了相应的属性。`getYearStart()`方法测试`$yearstart`属性是否已经设置了，如果还没有设置，`getYearStart()`把当前年份之前5年的值赋给`$yearstart`。`getYearEnd()`方法执行类似的操作。

随着程序清单24.6的继续，我们现在准备好创建这个类的最后一部分了。

程序清单24.6 创建一个日历库(续)

```

60:     function output() {
61:         if ($this->timestamp < 0) {
62:             $this->setDate_global();
63:         }
64:         $datearray = getdate($this->timestamp);
65:         $out = $this->day_select($this->name, $datearray);
66:         $out .= $this->month_select($this->name, $datearray);
67:         $out .= $this->year_select($this->name, $datearray);
68:         return $out;
69:     }
70:
71:     function day_select($fieldname, $datearray) {
72:         $out = "<select name=\"\$fieldname\".\"['mday']\">\n";
73:         for ($x=1; $x<=31; $x++) {
74:             $out .= "<option value=\"\$x\".\"(\$datearray['mday']==(\$x)";
75:             ?" selected":"").">".sprintf("%02d", $x) . "</option>\n";
76:         }
77:         $out .= "</select>\n";
78:         return $out;
79:     }
80:
81:     function month_select($fieldname, $datearray) {
82:         $out = "<select name=\"\$fieldname\".\"['mon']\">\n";
83:         for ($x = 1; $x <= 12; $x++) {
84:             $out .= "<option value=\"\".\"(\$x).\"\".\"(\$datearray['mon']==(\$x)";
85:             ?" selected":"")."> ".$this->months[$x-1]. "</option>\n";
86:         }
87:         $out .= "</select>\n";
88:         return $out;
89:     }
90:
91:     function year_select($fieldname, $datearray) {
92:         $out = "<select name=\"\$fieldname\".\"['year']\">";
93:         $start = $this->getYearStart();
94:         $end = $this->getYearEnd();
95:         for ($x= $start; $x < $end; $x++) {
96:             $out .= "<option value=\"\$x\".\"(\$datearray['year']==(\$x)";
97:             ?" selected":"").">\".$x.\"</option>\n";
98:         }
99:         $out .= "</select>\n";
100:         return $out;
101:     }
102: }

```

output()方法占据了这些代码的大部分（第60行到第69行）。它首先检查\$timestamp属性，除非setDate方法中的一个已经调用了，否则

`$timestamp`的值将会设置为-1，并且`setDate_global()`将被默认地调用。然后，时间戳传递给`getdate()`函数来构建一个日期数组，并且调用方法来产生每个下拉列表。

`day_select()`方法（第71行到第79行）只是构建一个HTML `select`元素，其中对于一个月中31个可能日期的每一个都有一个`option`元素。这个对象的当前日期存储在`$datearray`参数变量中，在构建元素设置相关的`option`元素的选择属性时使用。`sprintf()`函数格式化日期数，在日期1到9前添加0。`month_select()`和`year_select()`方法（第81行到第101行）使用类似的算法来构建月份和年份下拉列表。

为什么我们要把输出代码分解为4个方法，而不是简单地编写一个代码块呢？在构建一个类的时候，有两种程序员：一种是想要直接实例化一个`date_pulldown`对象，而另一个想要把`date_pulldown`类子类化以限制其功能。

对于前者，我们想要为类的功能提供一个简单而清晰的接口，然后这个程序员可以实例化一个对象，设置其日期，并且调用`output()`方法。对于后者，我们想要使得修改类的功能的离散元素变得容易。如果把所有输出代码放入到一个方法，将导致一个需要调整输出的子类复制大量完全可用的代码。通过把这些代码分解成离散的方法，我们就使得子类可以改变有限的部分功能，而不会打乱整体秩序。例如，如果一个子类需要把年份下拉列表表示为两个单选按钮，程序员只需要简单地覆盖`year_select()`方法。

程序清单24.7包含了一些代码，这些代码调用库类。在尝试执行这些代码之前，先取得来自程序清单24.6的代码，用PHP开始标记和结束

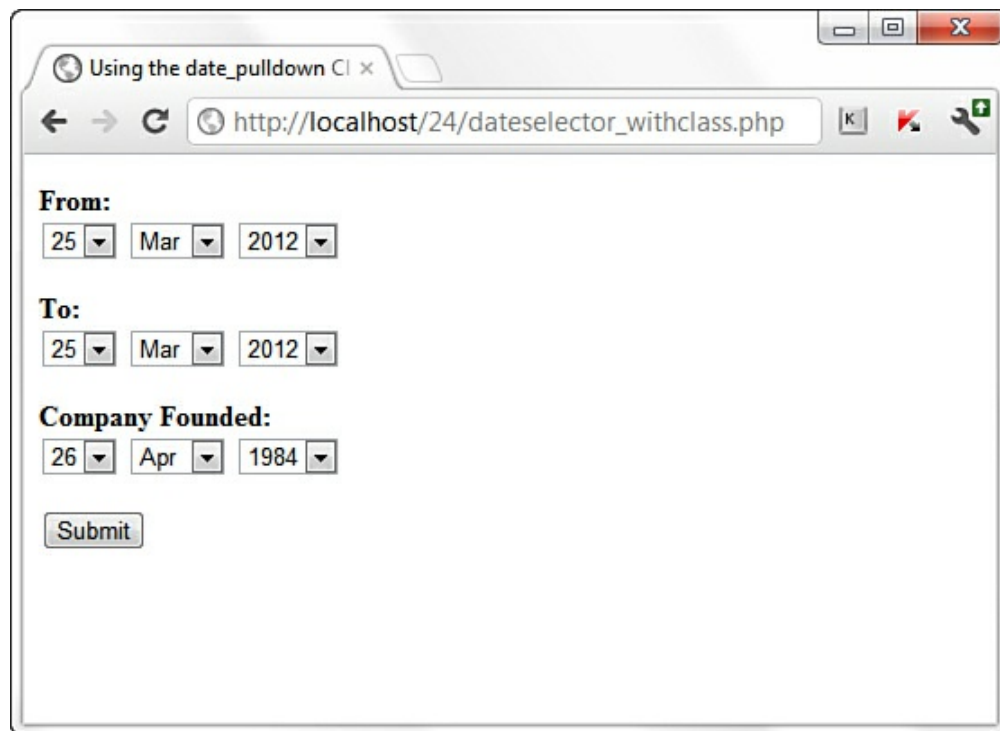
标记包围起来，然后保存到一个名为date\_pulldown.class.php的文件中。把这个文件放到Web服务器文档根目录下，因为程序清单24.7将会用到它，因此它最好位于那里。

程序清单24.7 使用date\_pulldown类

```
1:  <!DOCTYPE html>
2:  <html>
3:  <head>
4:  <title>Using the date_pulldown Class</title>
5:  </head>
6:  <?php
7:  include("date_pulldown.class.php");
8:  $date1 = new date_pulldown("fromdate");
9:  $date2 = new date_pulldown("todate");
10: $date3 = new date_pulldown("foundingdate");
11: $date3->setYearStart("1972");
12: if (empty($foundingdate)) {
13:     $date3->setDate_array(array('mday'=>26, 'mon'=>4, 'year'=>1984));
14: }
15: ?>
16: <body>
17: <form>
18: <p><strong>From:</strong><br/>
19: <?php echo $date1->output(); ?></p>
20:
21: <p><strong>To:</strong><br/>
22: <?php echo $date2->output(); ?></p>
23:
24: <p><strong>Company Founded:</strong><br/>
25: <?php echo $date3->output(); ?></p>
26:
27: <button type="submit" name="submit" value="submit">Submit</button>
28: </form>
29: </body>
30: </html>
```

在第6行，我们包含了date\_pulldown.class.php。包含了类文件之后，我们可以使用其所有的方法。对所有下拉列表都使用类的默认行为，除了“foundingdate”以外。对于这个特殊的对象，我们覆盖了默认的开始年份，在第10行将其设置为1972。在第12行，我们赋给这个下拉列

表一个任意的日期，这个日期直到表单提交的时候才会显示（参见图24-6）。



The screenshot shows a web browser window with the title "Using the date\_pulldown Cl x". The address bar displays "http://localhost/24/dateselector\_withclass.php". The form contains three sections: "From:" with date pickers for 25, Mar, and 2012; "To:" with date pickers for 25, Mar, and 2012; and "Company Founded:" with date pickers for 26, Apr, and 1984. A "Submit" button is located at the bottom of the form.

图24-6 date\_pulldown类所产生的下拉列表

**提示：**

这只是一个表单的前端，还没有操作或方法；我们需要提供自己的操作或方法以便使其真正地做些事情。

## 24.3 小结

在本章中，我们把在本书前面学习过的和日期相关的PHP函数组合到一起，开发了一个基本的日历显示应用程序。我们学习了如何使用 `checkdate()` 测试一个输入日期的有效性，并且通过一个示例脚本应用了已经学到的一些工具。我们还学习了在日历应用程序中添加和浏览事件的一种方法。我们还学习了如何构建一个日期相关的类库，这个类库可以用来使得HTML表单中操作日期的某些繁琐工作变得自动化。

## 24.4 Q&A

**Q:** 有很多函数用来在不同的日历之间转换吗？

**A:** 是的。PHP提供了适合不同日历的一整套函数。我们可以在位于<http://www.php.net/manual/en/ref.calendar.php> 的官方PHP手册中阅读到这些内容。

## 24.5 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。



## 问答题

1. 我们使用哪个PHP函数来创建一个时间戳？
2. 我们使用哪个PHP函数来创建一个和日期相关的信息的关联数组？
3. date\_pulldown类中的变量为何要声明为public的？

## 解答

1. `mktime()`
2. `getdate()`
3. `public`的变量在该类之外也可以使用，这些变量需要这样。

## 思考题

1. 修改日历显示脚本，以显示整个一年的日历，从1月到12月。然后，将日历显示为一个 $3 \times 4$ 的表格，或者4行，每行3个月。
2. 修改创建事件的脚本，以使用日期下拉类。

## 第25章 限制对应用程序的访问

在本章中，你将学到：

- 如何根据用户、客户机**IP**地址、域名和浏览器版本来限制访问。
- 如何使用**Apache**提供的用户管理工具。
- 如何存储和获取**cookie**信息。
- 如何使用**cookie**进行验证。

本章说明了如何使用**Apache**，根据用户的身份或请求的信息来限制对一个站点的访问。在应用程序中，我们可以创建自己的机制来进行用户验证以及通过**cookie**检查用户的有效性。

## 25.1 验证概览

授权和验证是很多站点经常需要的。首先，我们来依次了解一些定义。

验证建立了一次通讯中的参与者的身份。我们可以通过如下方式来获得验证：我们所知道的一个密码或者一个cookie，或者ID卡或钥匙这样现实的东西，或者像指纹或虹膜这样我们本身所固有的一部分，或者是这些元素的任意组合。在一个站点的环境中，验证通常是限制于使用密码和认证。

授权负责保护对资源的访问。我们可以根据几个因素来授权访问，例如，用户所来自的IP地址，用户的浏览器类型，用户试图访问的内容或者用户的身份（在前面通过验证来定义）。

Apache包含了几个模块用来提供验证和访问控制，并且它们可以用来保护动态的和静态的内容。我们可以使用这些模块中的一个，或者在应用程序级实现自己的访问控制并且提供自定义的登录界面、单点登录或者其他高级功能。

### 25.1.1 客户机验证

验证用户是为了记录或授权。HTTP规范提供了两种验证机制：基本验证和摘要验证。在这两种情况下，过程都是如下所示。

1. 客户机试图访问Web服务器上的限制内容。

2. Apache检查客户机是否提供了一个用户名和密码。如果没有，Apache返回一个HTTP 401状态码，表示需要用户验证。

3. 客户机读取到响应并且提示用户需要用户名和密码（通常使用一个弹出对话框）。

4. 客户机再次尝试访问Web页面，这次，把用户名和密码作为HTTP请求的一部分传送。客户机记住用户名和密码并且在随后对同一站点的请求中发送它们，这样，用户就不需要对每一个请求都再次输入用户名和密码了。

5. Apache检查身份的有效性，并且根据用户身份以及其他的访问规则来授权访问或拒绝访问。

在基本的验证模式中，用户名和密码都以明文传送，作为HTTP请求标头的一部分。这就暴露了一个安全风险，因为一个攻击者可以很容易地看到服务器和浏览器之间的对话，获知用户名和密码，并且此后自由地使用它们。

摘要验证提供了更强的安全性，因为它传送的是一个摘要而不是明文密码。摘要是根据几个参数的组合，这些参数包括用户名、密码和请求方法。服务器可以自行计算摘要并检查知道密码的客户机，即便密码本身并没有通过网络传送。

**提示：**

摘要算法是一种数学计算，它获取一个文本而返回另一个文本（即摘要），摘要唯一地标识了最初的文本。好的摘要算法应该确保（至少为了实用的目的）不同的输入文本产生不同的摘要，并且无法从摘要推导出最初的输入文本。MD5就是一种常用的摘要算法。

---

浏览器中对于摘要验证的支持有所不同，所以，你可能要将摘要验证限制使用于我们可以控制客户机的浏览器的情况下，例如，在公司的内部网中。

在任何情况下，对于摘要验证和基本验证，被请求信息本身未经保护地在网络上传送。Web站点的安全访问的一个更好的选择是在SSL协议上使用HTTP，我们将在第30章介绍。

### 25.1.2 用户管理方法

当验证模块接收到来自客户机的用户名和密码后，它需要根据已有的用户数据库来验证其有效性。用户名和密码可以用不同的方式存储，包括Apache所提供的文件方式和基于数据库的机制。第三方模块提供对于像轻量级目录访问协议（Lightweight Directory Access Protocol, LDAP）和网络信息服务（Network Information Services, NIS）等额外机制的支持。

## 25.2 Apache验证模块功能

Apache提供了基本的框架和命令来执行验证和访问控制。验证模块根据一个特定的后端方法（文件、数据库等）对验证密码提供支持。用户可以可选地组织成组，使访问控制规则更容易管理。

Apache提供了3种和使用任何验证模块的验证相关的、内建的指令：`AuthName`、`AuthType`和`Require`。

`AuthName`接受一个字符串参数，这是验证领域的名字。验证领域是Web服务器上我们可以向其询问密码的一个逻辑区域。它将会在浏览器弹出窗口中显示。

`AuthType`指定了浏览器验证类型，基本验证或摘要验证。

`Require`为我们指定一个允许访问的用户或组的列表。语法是`Require user`后面跟着一个或多个用户名，或者`Require group`后面跟着一个或多个组名，示例如下。

```
Require user joe bob
```

或者

```
Require group employee contractor
```

如果我们想要对提供了一个有效的用户名和密码的任何人授权访问，我们可以这样做：

```
Require valid-user
```



使用前面的指令，可以控制谁能够访问特定的虚拟主机、目录、文件等。尽管验证和授权都是独立的概念，实际上，它们在Apache中是联系在一起的。访问根据具体用户身份或者组成员来授权。一些第三方模块，例如，某些基于LDAP的模块，在验证和授权之间进行了清晰的划分。

包含在Apache中的验证模块提供了如下功能。

- 后端存储 —— 提供了包含用户名和组信息的文本或数据库文件。
- 用户管理 —— 提供了在后端存储中创建和管理用户和组的工具。
- 授权信息 —— 指定模块的结果是否是授权的。

**提示：**

有时不允许用户访问一个特定的领域，因为他们的信息没有在模块所提供的用户数据库中找到，或者因为没有匹配他们的信息的验证规则。在这个例子中，将会发生如下两种情况之一。

- 如果模块指定自己的结果为授权，用户将会被拒绝访问，并且Apache将会返回一个错误。
- 如果模块指定自己的结果为没有授权，其他模块可以有一个验证该用户的机会。

这就使得我们能够有一个主授权模块，它知道大多数用户，并且能够拥有其他的授权模块，这些模块可以验证其他的用户。

## 25.2.1 基于文件的验证

mod\_auth Apache模块提供了通过包含用户名和密码的文本文件进行的基本验证，类似于传统的UNIX验证如何使用/etc/passwd和/etc/groups文件。

## 1. 后台存储

当我们使用后台存储方法，你需要指定包含了用户名和密码的列表的文件，并且这个文件可选地包含了组的列表。

这个用户文件是一个UNIX式的密码文件，包含了用户的名字和加密的密码。在UNIX上，使用了加密算法后，整个看上去如下所示。

```
admin:iFrlxqg0Q6RQ6
```

在Windows系统，使用MD5算法，如下所示。

```
admin:$apr1$Ug3.....$jVTedbQWBKTfXsn5jK6UX/
```

组文件包含了一个组以及属于每个组的用户的列表，中间用空格隔开，像下面的条目一样。

```
web: admin joe Daniel
```

AuthUserFile和AuthGroupFile指令接受一个路径参数，该参数指向用户文件和组文件。组文件是可选的。

## 2. 用户管理

Apache发布版包括了UNIX上的htpasswd工具以及Windows上的htpasswd.exe，它们都设计用来帮助我们管理用户密码文件。两个版本的功能相同，但是Windows版本使用一种不同的方法来加密密码。加密对于用户和管理员是透明的。

在Linux/UNIX上，第一次添加一个用户的时候，我们需要输入如下命令。

```
/usr/local/apache2/bin/htpasswd -c file userid
```

这里file是包含用户名和密码列表的文件，而userid则是要添加的用户名。将会提示我们输入一个密码，并且将会创建文件。例如，在Linux/UNIX，执行如下命令。

```
/usr/local/apache2/bin/htpasswd -c /usr/local/apache2/conf/htusers admin
```

上述命令创建了密码文件/usr/local/apache2/conf/htusers，并且添加了admin用户。

类似的功能在Windows上也存在，这里的命令行操作看上去如下所示。

```
htpasswd -c "C:\Program Files\Apache Software Foundation\Apache2.2\conf\htusers" admin
```

-c命令行选项告诉htpasswd可执行文件，它应该创建文件。当我们想要把用户添加到一个已有的密码文件中时，不要使用-c选项，否则文件可能被覆盖。

把密码文件存储在文档根目录之外从而使得无法通过Web浏览器访问它，这一点很重要。否则，攻击者可以下载这个文件并且获取用户名和密码的列表。尽管密码是加密的，当你拥有这个文件，还是可能执行蛮力攻击来尝试猜出密码。

### 3. 使用mod\_auth

程序清单25.1给出了一个配置示例，它限制htusers密码文件中的授权用户访问文档根目录下的私有目录。注意，可选的AuthGroupFile指令并没有给出。

---

```
1: <Directory /usr/local/apache2/htdocs/private>
2: AuthType Basic
3: AuthName "Private Area"
4: AuthUserFile /usr/local/apache2/conf/htusers
5: Require valid-user
6: </Directory>
```

---

## 25.2.2 基于数据库文件的访问控制

以纯文本文件存储用户名和密码非常方便，但是这种方法缺乏灵活性。Apache将需要顺序地打开并读取文件，以查找某个特定用户。当用户的数量增加，这一操作将变得非常耗时。mod\_auth\_dbm模块使得我们可以用索引的数据库文件来替代基于文本的文件，这样可以处理更多数量的用户而不会导致性能下降。mod\_auth\_dbm模块包含在Apache中，但默认不可用。当构建Apache配置的时候，使用--enable-module=dbm选项就可以打开这一模块。

### 1. 后端存储

mod\_auth\_dbm模块提供了两个指令，AuthDBMUserFile和AuthDBMGroupFile，它们指向了包含用户名和组的数据库文件。和纯文本文件不同，这两条指令可以指向同一个文件，其中既包含用户也包含组。

### 2. 用户管理

Apache提供了一种UNIX和Windows用户都可以使用的工具，叫做htdbm，它允许你创建和管理存储在数据库文件中的用户和组。可以在Apache发布的bin目录下找到这个工具，一旦找到了，输入如下内容，向新数据库添加一个用户。

```
htdbm -c databasename userid
```

然后会提示你输入密码，并且将该用户添加到新的数据库文件中。  
如果你需要删除用户 `daniel`，可以执行如下的命令。

```
htdbm -x dbname daniel
```

可以通过执行不带任何参数的 `htdbm`，从而得到该命令的完整的语法信息。

## 25.3 使用Apache进行访问控制

`mod_access`模块默认可用，允许我们根据客户机请求的参数，例如一个具体标头、IP地址或客户机的主机名的出现，来限制对资源的访问。

### 25.3.1 实现访问规则

我们可以使用`Allow`和`Deny`指令来指定访问规则。这两个指令都接受IP地址、环境变量和域名这样的参数列表。

#### 1. 通过IP地址允许/拒绝访问

我们可以根据客户机的IP地址来拒绝或授权访问，示例如下。

```
Allow from 10.0.0.1 10.0.0.2 10.0.0.3
```

我们可以用一个部分的IP地址或者一个网络/掩码对来指定IP地址的范围。另外，我们可以指定IP地址的第一个、第二个或第三个字节。包含这些的任何IP地址都符合这一规则，示例如下。

```
Deny from 10.0
```

上述规则将会匹配所有以10.0打头的地址，如10.0.1.0和10.0.0.1。

我们也可以利用IP地址和网络掩码，IP地址指定了网络，而掩码指定了哪些位属于网络前缀而哪些位属于节点，示例如下。

```
Allow from 10.0.0.0/255.255.255.0
```

将会匹配IP地址10.0.0.1、10.0.0.2一直到10.0.0.254。

我们可以通过给出最高位为1的数量来指定网络掩码。例如，我们可以把上述的规则改写如下。

```
Allow from 10.0.0.0/24
```

## 2. 通过域名允许/拒绝访问

我们可以根据指定主机名或者部分域名来控制访问。例如，`Allow from example.com`将会匹配`www.example.com`，`foo.example.com`等。

提示：

根据域名打开访问规则迫使Apache在客户机地址上反向查找DNS，而忽略掉`HostNameLookups`指令的设置。这个过程是隐式地执行的。

## 3. 根据环境变量允许/拒绝访问

你可以根据出现某个环境变量来指定访问规则，通过在变量名的前面加上字符串`env=`前缀。我们可以使用这一功能来授权或拒绝访问某个浏览器或者浏览器版本，以防止某个站点链接到资源等等。为了让这个例子像期待的那样工作，客户机需要传送`User-Agent`标头，示例如下。

```
BrowserMatch MSIE iexplorer  
Deny from env=iexplorer
```

由于客户机发送了`User-Agent`标头，它有可能被忽略或者被操作，但是大多数用户不会这么做，并且这一技术在大多数情况下是有效的。

## 4. 允许/拒绝所有客户机访问

关键字`all`匹配所有客户机。我们可以指定`Allow from all`或`Deny from all`来授权或拒绝所有客户机访问。

## 25.3.2 应用访问规则

我们可以有数个Allow和Deny访问规则。我们可以通过Order指令来选择应用访问规则的顺序。后应用的规则具有更高的优先级。规则接收一个参数，这个参数可以是Deny，Allow、Allow，Deny或Mutual-Failure。Deny，Allow是Order指令的默认值。注意，值中没有空格。

### 1. Deny， Allow

Deny， Allow指定了Deny命令是在Allow指令之前应用的。通过Deny， Allow， 如果没有Allow或Deny指令或者客户机没有匹配到任何规则， 客户机默认被授权访问。如果客户机匹配一条Deny规则， 它将被拒绝访问， 除非它也匹配了一条Allow规则， 该Allow规则具有优先权， 因为Allow指令在后面应用， 并且具有更高的优先级。

程序清单25.2展示了如何配置Apache以允许来自内部网络或者域example.com的客户机访问/private位置， 而拒绝其他的任何人访问。

程序清单25.2 示例Deny， Allow访问控制配置

```
1: <Location /private>
2: Order Deny,Allow
3: Deny from all
4: Allow from 10.0.0.0/255.255.255.0 example.com
5: </Location>
```

### 2. Allow， Deny

Allow， Deny指定了Allow指令在Deny指令之前应用。通过Allow,Deny， 如果没有Allow或Deny指令或者如果客户机没有匹配任何规则， 客户机默认被拒绝访问。如果客户机匹配任何Allow规则， 将会允许它访问， 除非它也匹配了一条Deny规则， Deny规则将具有优先



权。

注意，不帶有任何Allow或Deny规则的Order Allow，Deny的出现，会导致对某个资源的所有请求被拒绝，因为默认的行为是拒绝访问。

程序清单25.3允许访问一个特定主机之外的任何主机。

程序清单25.3 示例Allow,Deny访问控制配置

```
1: <Location /some/location/>
2: Order Allow,Deny
3: Allow from all
4: Deny from host.example.com
5: </Location>
```

---

### 3. Mutual-Failure

在Mutual-Failure的情况下，只有当主机匹配一条Allow指令并且不匹配任何的Deny指令的时候，才授权主机访问。

## 25.4 组合Apache访问方法

在上一节中，我们学习了如何根据用户身份或请求信息来限制访问。**Satisfy**指令允许我们确定，为了授权访问是否两种访问类型限制必须按顺序满足。**Satisfy**接收一个参数，要么是**all**，要么是**any**。

**Satisfy all**意味着如果客户机提供了一个有效的用户名和密码并且通过了访问限制，客户机将被授权访问。**Satisfy any**意味着如果客户机提供一个有效的用户名和密码或者通过了访问限制，客户机将被授权访问。

为什么这条指令有用呢？例如，你可能想要为来自内部的、可信的地址的用户提供对Web站点的自由访问，但需要那些来自Internet的用户提供一个有效的用户名和密码。程序清单25.4展示了这一点。

程序清单25.4 混合的验证和访问控制规则

```
1: <Location /restricted>
2: Allow from 10.0.0.0/255.255.255.0
3: AuthType Basic
4: AuthName "Intranet"
5: AuthUserFile /usr/local/apache2/conf/htusers
6: Require valid-user
7: Satisfy any
8: </Location>
```

### 提示：

根据连接或请求信息的访问控制并不完全安全。尽管它在大多数情况提供了一个相应级别的保护，但规则依赖于你的DNS服务器的完整性和网络基础设施。如果一个攻击者获取了对DNS服务器的控制，或者路由器或防火墙配置不正确，他可以轻松地把授权域名记录修改为指向他自己的机器，或者假装他来自一个授权的IP地址。



## 25.5 根据HTTP方法限制访问

通常，我们希望访问控制指令应用于所有类型的客户请求，并且这是一个默认的行为。然而，在某些情况下，我们只想对某些HTTP方法，如GET和HEAD，应用验证和访问规则。

<Limit>容器接受方法的一个列表，并且包含了指令，这些指令应用于包含那些方法的请求。可用的方法的完整的列表是：GET、POST、PUT、DELETE、CONNECT、OPTIONS、TRACE、PATCH、PROPFIND、PROPPATCH、MKCOL、COPY、MOVE、LOCK和UNLOCK。

<LimitExcept>段提供了一个补充功能，它所包含的命令应用于那些不包含所列出的方法的请求。

程序清单25.5给出了默认的Apache配置文件的一个例子。<Limit>和<LimitExcept>段允许只读的方法，但是拒绝对任何可以修改文件系统内容的其他方法（如PUT方法）的请求。要了解有关这里可用的多种选项的更多信息，请参考位于<http://httpd.apache.org/docs-2.2/mod/core.html>的Apache文档。

程序清单25.5 根据规则限制访问

```
1: <Directory /home/*/public_html>
2:   AllowOverride FileInfo AuthConfig Limit
3:   Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
4:   <Limit GET POST OPTIONS PROPFIND>
5:     Order Allow,Deny
6:     Allow from all
7:   </Limit>
8:   <LimitExcept GET POST OPTIONS PROPFIND>
9:     Order Deny,Allow
10:    Deny from all
11:   </LimitExcept>
12: </Directory>
```

在下一节中，我们还将要学习根据cookie中的信息在应用程序中限制访问。

## 25.6 根据cookie值限制访问

在第12章，我们学习了cookie的所有结构，以及在PHP中如何设置和访问cookie变量。下面几节将介绍cookie在验证过程中的一些实际应用。

假设你已经创建了一个登录表单，用来根据数据库检查值。如果用户是授权的，我们发送一个cookie来说明这一点。随后，对于仅限于授权的用户访问的所有页面，我们检查这个具体的cookie。如果存在cookie，用户可以查看页面。如果不存在cookie，用户要么送回到登录表单，要么发送一条显示于屏幕上的关于访问限制的消息。我们将在下面的小节中经历这些步骤。

### 25.6.1 创建授权用户表

当我们要把用户账户集成到一个基于Web的应用程序中时，把与用户相关的信息存储到一个数据库表中，这是常见的做法。随后，这个表中的信息可以用来授权用户并且授予这些“特殊”用户对专门的站点区域的访问。

如下的表创建命令将会在你的MySQL数据库中创建一个名为auth\_users的表，其中具有ID、名字、姓氏、Email地址、用户名和密码的字段。

```
CREATE TABLE auth_users (
    id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
    f_name VARCHAR(50),
    l_name VARCHAR(50),
    email VARCHAR(150),
    username VARCHAR(25),
    password VARCHAR(41)
);
```

如下的INSERT命令在auth\_users table表中为名为John Doe的用户插入了一条记录，其Email地址为john@doe.com，用户名为jdoe，而密码为doepass。

```
INSERT INTO auth_users VALUES ('1', 'John', 'Doe', 'john@doe.com',
'jdoe', PASSWORD('doepass'));
```

这条INSERT命令应该是一目了然的，除了PASSWORD()函数的用法。当这个函数用在INSERT命令中时，存储在表中的实际上不是真正的密码，而是密码的一个41字符的哈希值。

当我们查看auth\_users表的内容的时候，我们会在password字段中看到哈希值，如下所示。

username	password
jdoe	*0AAD744979343D58A7F17A50E514E6AD6533D04B

尽管它看上去像是加密的，哈希值实际上不是信息的一个加密数据。实际上，它是原始信息的一个“指纹”。哈希值通常用来执行匹配，就像指纹一样。在这个例子中，当你要检查自己的用户密码的时候，将使用输入的哈希值来匹配存储的哈希值。使用哈希值避免了存储实际密码，也避免了安全风险。

## 25.6.2 创建登录表单和脚本

在授权了表中的用户之后，我们需要给他们一种方法来证明身份。在这个例子中，是一个简单的两字段的表单，如程序清单25.6所示。

程序清单25.6 用户登录页面

```
1:  <!DOCTYPE html>
2:  <html>
3:  <head>
4:  <title>User Login Form</title>
5:  </head>
6:  <body>
7:  <h1>Login Form</h1>
8:  <form method="post" action="userlogin.php">
9:  <p><label for="username"><strong>username:</strong><br/>
10: <input type="text" id="username" name="username" /></label></p>
11: <p><label for="password"><strong>password:</strong><br/>
12: <input type="password" id="password" name="password" /></label></p>
13: <button type="submit" name="submit" value="login">Login</button>
14: </form>
15: </body>
16: </html>
```

把这些代码放入到一个名为loginform.html的文本文件中，并且将这个文件放置到Web服务器文档根目录下。接下来，我们将创建脚本本身，表单脚本名为userlogin.php（参见程序清单25.7）。

程序清单25.7 用户登录脚本

```
1:  <?php
2:  //check for required fields from the form
3:  if ((!isset($_POST['username'])) || (!isset($_POST['password']))) {
4:      header("Location: userlogin.html");
5:      exit;
6:  }
7:
8:  //connect to server and select database
9:  $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB")
10:      or die(mysqli_error());
11:
12:  //use mysqli_real_escape_string to clean the input
13:  $username = mysqli_real_escape_string($mysqli, $_POST['username']);
14:  $password = mysqli_real_escape_string($mysqli, $_POST['password']);
```



```

15:
16: //create and issue the query
17: $sql = "SELECT f_name, l_name FROM auth_users WHERE
18:         username = '". $username.'" AND
19:         password = PASSWORD('". $password."')";
20: $result = mysqli_query($mysqli, $sql) or die(mysqli_error($mysqli));
21:
22: //get the number of rows in the result set; should be 1 if a match
23: if (mysqli_num_rows($result) == 1) {
24:
25:     //if authorized, get the values of f_name l_name
26:     while ($info = mysqli_fetch_array($result)) {
27:         $f_name = stripslashes($info['f_name']);
28:         $l_name = stripslashes($info['l_name']);
29:     }
30:
31:     //set authorization cookie
32:     setcookie("auth", "1", 0, "/", "yourdomain.com", 0);
33:
34:     //create display string
35:     $display_block = "
36:     <p>". $f_name." ". $l_name." is authorized!</p>
37:     <p>Authorized Users' Menu:</p>
38:     <ul>
39:     <li><a href=\"secretpage.php\">secret page</a></li>
40:     </ul>";
41: } else {
42:     //redirect back to login form if not authorized
43:     header("Location: userlogin.html");
44:     exit;
45: }
46: //close connection to MySQL
47: mysqli_close($mysqli);
48: ?>
49: <!DOCTYPE html>
50: <html>
51: <head>
52: <title>User Login</title>
53: </head>
54: <body>
55: <?php echo $display_block; ?>
56: </body>
57: </html>

```

把上述代码放入到一个名为userlogin.php的文本文件中，在第32行

把“your-domain.com”修改为你实际的域名，并且把该文件放置到Web服务器文档根目录下。此时，你可以尝试进行一下，但先来看看代码将做些什么。

第3行检查两个必需的字段，也就是表单中仅有的两个字段：`$_POST['username']`和`$_POST['password']`。如果这些字段中的任何一个不存在，脚本都把用户重定向到最初的登录表单。如果两个字段都存在，脚本跳到第9行，从那里连接到数据库服务器，以准备执行SQL查询来检查用户的身份。在执行查询之前，第13行和第14行将用户输入的数据安全化。一旦信息安全化了，执行查询，可以在第17行到第20行找到。注意，查询根据存储在表中的密码，检查了来自表单输入的密码的哈希值。

这两个元素必须相互匹配，并且都属于所涉及的用户名，也就是被授权的用户。

第23行通过计算结果集中的行数来测试查询的结果。如果用户名和密码对代表一个有效的登录，行数计算应该确切地为1。如果是这种情况，在第26行到第29行，`mysqli_fetch_array()`函数用来提取用户的名字和姓氏。这些名字只是为了美观。

第32行设置了授权cookie，这个cookie的名字是auth，并且值为1。如果在这个时间空档放入一个0，只要这个用户的Web浏览器会话打开着，cookie就会持续。当用户关闭了浏览器，cookie将过期。第35行到第40行创建了一条供显示的消息，包括到一个文件的链接，我们稍后将创建这个文件。最后，在第41行到第45行处理一个失效的登录尝试。在这个例子中，用户被直接重定向到最初的登录表单。

访问登录表单，并且为John Doe用户输入有效的值。当你提交了表单，结果如图25-1所示。

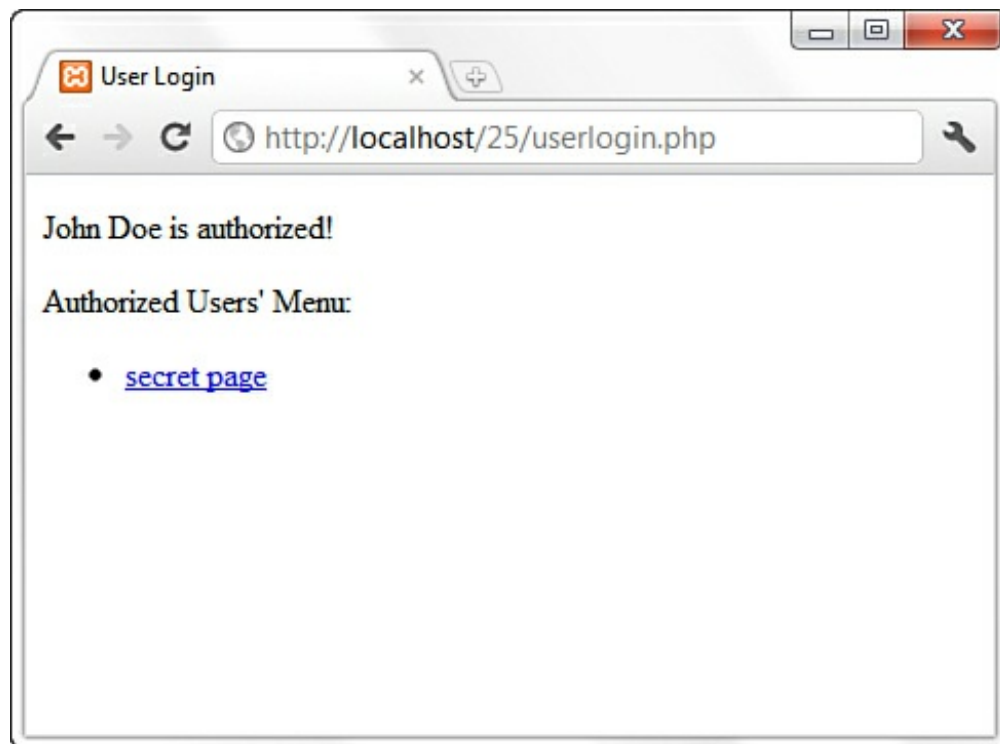


图25-1 成功的登录结果

尝试用一个无效的用户名和密码对登录，你应该被重定向到登录表单。在下一节（也是最后一节）中，我们将创建secretpage.php脚本，它将读取我们刚设置的验证cookie并进行相应的操作。

### 25.6.3 测试auth cookie

这个问题的最后一部分就是使用auth cookie的值，从而允许用户访问一个私有文件。在这个例子中，所涉及的文件在程序清单25.8中给出。

程序清单25.8 检查auth cookie

```
1: <?php
2: if ($_COOKIE['auth'] == "1") {
3:     $display_block = "<p>You are an authorized user.</p>";
4: } else {
5:     //redirect back to login form if not authorized
6:     header("Location: userlogin.html");
7:     exit;
8: }
9: ?>
10: <!DOCTYPE html>
11: <html>
12: <head>
13: <title>Secret Page</title>
14: </head>
15: <body>
16: <?php echo $display_block; ?>
17: </body>
18: </html>
```

在图25-1所示的菜单中，点击secret page链接。由于你是一个被授权的用户，应该看到如图25-2所示的结果。

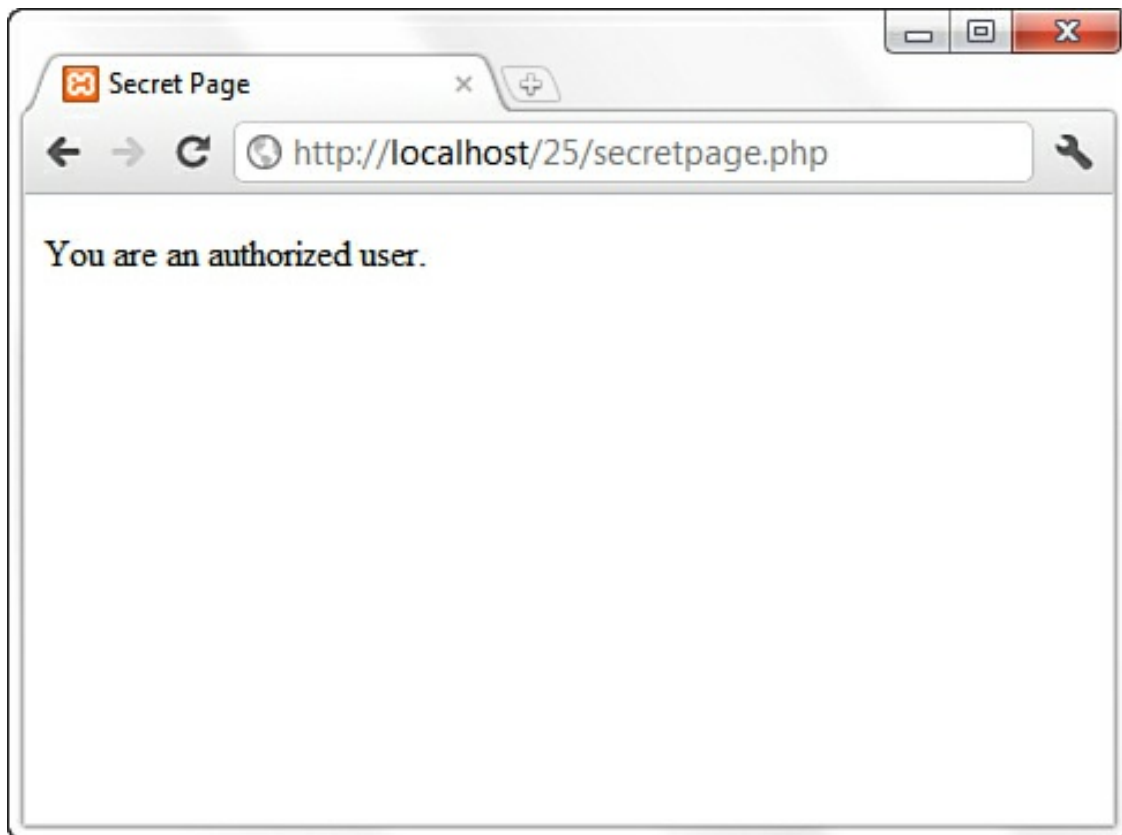


图25-2 作为一个授权用户访问secret page

关闭浏览器并且尝试直接访问secretpage.php。你将会发现，无法直接访问，并且将会重定向到最初的登录页面，因为验证cookie还没有设置。

## 25.7 小结

本章介绍了如何使用Apache功能来根据远程用户的身份，以及来自HTTP请求或网络连接的信息限制对Web站点的访问。本章还介绍了一些验证模块，包括Apache和其他工具，这些模块和工具可以用来创建并管理用户和组数据库。

此外，我们学习了使用cookie值来允许访问PHP应用程序的特定部分的一种方法。

## 25.8 Q&A

**Q:** 我已经有一个**UNIX**系统，可以使用**/etc/passwd**作为用户数据库吗？

**A:** 尽管使用**/etc/passwd**可能看上去很方便，但还是建议你不要使用已有的**/etc/passwd**文件来验证站点的用户。否则，获取了对Web站点的用户的访问权限的攻击者，将能够获取对系统的访问权限。保持数据库隔离并且鼓励用户为系统账户和Web访问设置不同的密码。对于安全性较弱的密码以及用户名就是密码的账户，定期运行密码检查器来扫描它们。

**Q:** 为什么在某些**Web**站点中需要两次密码？

**A:** 浏览器记录了密码，以便我们不用在每次请求的时候都必须输入密码。存储的密码基于领域（**AuthName**命令）以及Web站点的主机名。有时候，我们可以通过不同的名字（如**yourdomain.com**和**www.yourdomain.com**）来访问一个Web站点。如果授权你访问**yourdomain.com**的某一个限制区域，但被重定向或者打开到**www.yourdomain.com**的链接，将会再次要求提供用户名和密码，因为浏览器认为这是一个完全不同的Web站点。

**Q:** **cookie**是否会引起任何严重的安全或隐私问题？

**A:** 服务器访问的cookie只能是从自己的域中设置的。尽管一个cookie可以存储在用户的硬盘上，但没有对用户文件系统的其他访问权限。然而，有可能设置一个cookie来响应对一个图像的请求。因此，如果很多站点包含了一个第三方广告服务器或计数器脚本所提供的图像，第三方可能能够跨越多个域来跟踪一个用户。



## 25.9 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 相对于纯文本文件，用来存储用户验证信息的数据库文件的优势是什么？
2. 能否说出HTTP基本验证的一些缺点？
3. 哪个函数设计用来允许我们在访问者的浏览器上设置一个cookie？

## 解答

1. 数据库文件具有更好的可扩展性，因为它们可以索引。这意味着Apache不需要顺序读取文件直到找到针对一个特定用户的匹配，而是可以跳到准确的位置。

2. 一个缺点是，信息在网络上以明文传输。这意味着，除非你使用SSL，否则攻击者有可能读取到浏览器发送到服务器的数据包并且窃取你的密码。另一个缺点是，HTTP验证不会提供一个工具来自定义登录（除了领域名）。Web站点使用HTML表单和cookie来实现自定义登录机制是很常见的。

3. `setcookie()`函数允许你设置一个cookie，尽管你也可能使用`header()`函数输出一个Set Cookie标头。

## 思考题

练习在你的开发服务器上使用不同类型的验证，既包括基于服务器的验证，也包括使用PHP的验证。体会基本的HTTP验证和自己设计的验证之间的差别。

## 第26章 记录并监视Web服务器活动

在本章中，你将学到：

- 如何理解**Apache**日志格式和日志级别。
- 如何备份和分析**Apache**日志。
- 如何解释可能出现在日志中的常见错误。
- 如何创建把指定项目记录到数据库表的脚本。
- 如何创建基于这些日志表的自定义报告。

本章介绍**Apache**中的日志系统如何工作，以及如何自定义日志系统，包括要存储哪些信息以及信息存储在哪里。另外，我们还将学习一种快捷方法，以使用**PHP**和**MySQL**来记录**Apache**日志文件领域之外你感兴趣的内容。

## 26.1 标准Apache访问日志

使用Apache的基本日志功能，我们可以通过记录对宿主Web站点的服务器的访问，来记录谁访问了Web站点。我们可以记录浏览器请求和服务器相应的每个方面，包括客户机的IP地址、用户以及所访问资源。要创建一个请求日志，我们需要如下3个步骤。

1. 定义想要记录什么——日志格式。
2. 定义将其记录在何处——日志文件、一个数据库、一个外部程序。
3. 定义是否记录——条件式的日志规则。

下面几节进一步讨论这些步骤。

### 26.1.1 确定记录什么

尽管日志几乎涉及到和请求相关的每个方面，我们可以通过创建一个日志格式来定义日志条目如何出现。日志格式就是一个字符串，其中包含了和日志格式化指令混合的文本。日志格式化指令以a%打头，并且后面跟着一个指令名或标识符，通常是表示要记录的信息片断的一个字母。

当Apache记录一个请求的时候，它扫描这个字符串并且用值替代每个指令。例如，如果日志格式是This is the client address %a，日志条目就是类似This is the client address 10.0.0.2的内容。也就是说，日志指

令[%a](#)由发出请求的客户机的IP地址所取代。表26-1提供了所有格式化指令的一个完整列表。

表26-1 日志格式化指令

格式化选项	说 明
来自客户机的数据	
<a href="#">%a</a>	来自客户机的远程IP地址
<a href="#">%h</a>	发出请求的客户机的主机名或IP地址。是否记录主机名取决于两个因素：客户机的IP地址必须通过一个反向DNS查找解析为一个主机名，并且Apache必须配置为使用HostNameLookups指令来执行此查找；后者将在本章后面介绍。如果这两个条件不满足，将记录客户机的IP地址而不是主机名
<a href="#">%l</a>	远程用户，通过identd协议获取。这个选项并不是非常有用，因为在大多数客户机上，并不支持这个协议
<a href="#">%u</a>	远程用户，来自HTTP基本验证协议
来自服务器的数据	
<a href="#">A%</a>	来自服务器的本地IP地址
<a href="#">%D</a>	服务请求所花的时间，以微秒为单位
<a href="#">%{env_variable}e</a>	一个名为env_variable的环境变量的值（这里有很多个）
<a href="#">%{time_format}t</a>	当前时间。如果{time_format}存在，它将会解释为UNIX strftime函数的一个参数。参见Apache手册的logresolve页面了解详细内容
<a href="#">%T</a>	服务请求所花的时间，以秒为单位
<a href="#">%v</a>	应答请求的服务器的规范的名称
<a href="#">%V</a>	根据UseCanonicalName指令的服务器名称
<a href="#">%X</a>	到服务器的连接的状态。值为x的时候，意味着在服务器可以发送数据之前，放弃连接。A+意味着对于来自同一客户机的进一步请求保持连接可用。A-意味着连接将被关闭
来自请求的数据	
<a href="#">%{cookie_name}C</a>	一个名为cookie_name的cookie的值
<a href="#">%H</a>	请求协议，如HTTP或HTTPS
<a href="#">%m</a>	GET、POST、PUT等请求方法
<a href="#">%{header_name}i</a>	来自客户机的请求中，名为header_name的一个标头的值。这个信息可能很有用，例如，要记录访问者的浏览器的名字和版本
<a href="#">%r</a>	最初的HTTP请求的文本

%q	查询参数，如果有的话，以a?为前缀
%U	请求的URL，没有查询参数
%y	用于HTTP验证（基本的或摘要的）的用户名来自响应的数据
%b、%B	发回给客户机的响应的主体大小（不包括标头），以字节为单位。两个选项之间的唯一区别是，如果没有数据发回，%b将记录一个-，而%B将记录一个0
%f	服务的文件的路径，如果有文件的话
%t	请求得到服务的时间
% {header_name}o	对客户机的响应中名为header_name的一个标头的值
%>s	最终状态代码。Apache可以数次处理同一个请求（内部重定向）。这是最终响应的状态代码

通用日志格式（Common Log Forma，CLF）是一个标准日志格式。大多数Web站点可以使用这一格式记录请求，并且，这一格式为许多日志处理和报表工具所理解，其格式如下所示。

```
"%h %l %u %t \"%r\" %>s %b"
```

也就是说，它包括客户机的主机名或IP地址。通过identd的远程用户，通过HTTP验证的远程用户，请求得到服务的时间、请求的文本、状态代码以及服务的内容的字节大小。

#### 提示：

我们可以阅读位于<http://www.w3.org/Daemon/User/Config/Logging.html> 的最初的W3C服务器的通用日志格式文档。

如下是一个示例的CLF条目。

```
10.0.0.1 - - [19/Jan/2012:17:32:43 -0500] "GET / HTTP/1.0" 200 1101
```

现在我们准备好学习如何使用LogFormat指令定义日志格式。这个



指令接受两个参数：第一个参数是日志字符串，而第二个参数是将来和日志字符串关联的一个别名。

例如，来自默认Apache配置文件的如下指令定义了CLF并且为其分配一个别名common。

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

你也可以使用只带一个参数的LogFormat指令，这个参数要么是一个日志格式字符串，要么是一个别名。这将会具有为TransferLog指令所使用的日志格式设置默认值的效果，我们将在本章稍后介绍这一点。

## 1. HostNameLookups指令

当一个客户机做出请求，Apache只知道客户机的IP地址。Apache必须执行所谓的反向DNS查询来得出和这个IP地址相关的主机名。这个操作可能颇费时间并且可能导致请求处理的显著的延迟。

HostNameLookups指令允许我们控制是否执行反向DNS 查询。

HostNameLookups指令可以接受如下参数中的一个：on、off或double，默认值是off。double查询参数意味着Apache将通过IP找到主机名，并且随后尝试从主机名找到IP。正如<http://httpd.apache.org/docs-2.0/dns-caveats.html> 所介绍的，如果我们真的关心安全性，这个过程是需要的。如果我们要使用主机名作为Allow和Deny规则的一部分，应该执行一个double DNS查询而不管HostNameLookups的设置。

如果HostNameLookups是打开的（on或者double），Apache将会记录主机名。这会增加服务器上的额外负担，当我们决定打开或关闭HostNameLookups的时候，应该意识到这一点。如果我们选择保持

HostNameLookups关闭，这对于中高流量的站点来说是推荐的做法，Apache将只记录相关的IP地址。随后，有众多的工具可以用来解析日志里的IP地址。请参阅本章中的“管理Apache日志”一节。此外，结果将通过环境变量REMOTE\_HOST传递给CGI脚本。

## 2. IdentityCheck指令

在本章开始处，介绍了如何通过identd协议使用%l日志格式化指令来记录远程用户名。IdentityCheck指令接受一个on或off的值，来打开或关闭对这个值的检查并使其可供日志中的内容使用。由于信息并不可靠并且要花很长时间去检查，默认情况下，这个选项关闭并且不可用。我们提到%l，只是因为它是CLF的一部分。要了解identd协议的更多内容，可以参考位于<http://www.rfc-editor.org/rfc/rfc1413.txt> 的RFC 1413。

## 3. 状态码

我们可以指定是否在日志条目中记录特定的元素。在本章开始，我们学习了以a%开始的日志指令，其后跟着一个指令标识符。在这之间，我们可以插入一个状态码的列表，中间用逗号隔开。如果请求状态是列出的代码中的一个，将记录该参数，否则将记录-。

例如，如下的指令标识符将记录错误请求（状态码400）以及带有未实现的方法的请求（状态码501）的浏览器名称和版本。这些信息对于记录哪个客户机引发了问题很有用。

```
%400,501{User-agent}i
```

我们可以在方法列表的前面加上一个“!”，表示如果方法没有实现，就记录参数。

```
%!400,501{User-agent}i
```

## 26.1.2 记录对文件的访问

记录文件是Apache中记录请求的默认方式。我们可以使用TransferLog和CustomLog指令来定义文件名。TransferLog指令接受一个文件参数并且使用LogFormat指令所定义的最新的日志格式，LogFormat指令只带有一个参数（别名或格式字符串）。如果没有日志格式存在，默认值为CLF。

下面的例子展示了如何使用LogFormat和TransferLog指令来定义一个日志格式，该日志格式以CLF为基础但也包含浏览器名称。

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{User-agent}i\""
TransferLog logs/access_log
```

CustomLog指令允许我们显式地指定日志格式。它至少接受两个参数：一个日志格式以及一个目标文件。日志格式可以作为别名指定，或者直接指定为一个日志字符串。

指令示例如下。

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{User-agent}i\"" myformat
CustomLog logs/access_log myformat
```

上述指令和如下指令是等效的。

```
CustomLog logs/access_log "%h %l %u %t \"%r\" %>s %b \"%{User-agent}i\""
```

CustomLog指令接受一个环境变量作为第三个参数。如果这个环境变量存在，条目将记录，否则将不会记录。如果环境变量通过一个“!”前缀求反，如果变量不存在，将记录条目。

下面的例子展示了如何避免在日志中记录GIF和JPEG格式的图像。

```
SetEnvIf Request_URI "(\\.gif|\\.jpg)$" image
CustomLog logs/access_log common env=!image
```

提示：

在httpd.conf文件的此处和其他区域中用于模式匹配的正则表达式，和PHP以及其他编程语言中的正则表达式遵从同样的格式。

### 26.1.3 记录对一个程序的访问

TransferLog和CustomLog指令都可以接受一个以管道符|为前缀的可执行程序作为一个参数。Apache将会把日志条目写入程序的标准输入。反过来，这个程序将会处理输入，例如把条目记录到数据库，或将它们传送到另一个系统等。

如果该程序由于某些原因而停止，服务器将确保它重新启动。如果服务器死机，程序也会停止。绑定到Apache的rotatelogs工具，就是日志程序的一个例子，我们将在本章后面介绍它。作为一个通用的规则，除非有了使用一个特定程序的具体需求，否则很容易并且更可能记录到磁盘上的一个文件，并且随后可能在另外一台不同的机器上进行处理、合并、日志分析等。

确保用来记录请求的程序是安全的，因为它作为Apache所启动的一个用户运行。在UNIX上，这通常意味着root，因为外部程序在服务器改变将其用户ID改为User指令的值之前启动，通常User指令的值为nobody或www。

## 26.2 标准Apache错误日志

除了记录客户请求，Apache还可以配置为记录错误信息以及调试信息。除了Apache自身产生的错误，CGI错误也可以记录。

每个错误日志条目前面都会有错误发生的时间以及客户机的IP地址或主机名，如果有这些信息的话。和HTTP请求记录一样，我们可以把错误信息记录到一个文件或程序中。在UNIX系统上，我们也可以记录到syslog守护进程。在Windows上，错误可以记录到Windows事件日志并且可以通过Windows事件查看器查看。使用ErrorLog指令来定义要记录到哪里。

### 26.2.1 把错误记录到一个文件

一个文件参数给出了错误日志文件的路径。如果这个路径是相对的，假设它相对于服务器根目录。默认情况下，错误日志文件位于logs目录下，并且在UNIX上名为error\_log，在Windows上名为error.log。下面是一个例子。

```
ErrorLog logs/my_error_log
```

### 26.2.2 把错误记录到一个程序

我们可以指定到一个程序的路径，用管道符“|”作为前缀。Apache将会把错误记录到该程序的标准输入中，并且，该程序会进一步处理它们。下面是一个例子。

```
ErrorLog "|/usr/local/bin/someprogram"
```

### 26.2.3 syslog守护进程参数

在UNIX系统上，如果我们指定了syslog作为一个参数，可以把错误消息记录到UNIX系统日志守护进程syslogd。默认情况下，日志错误记录到syslog工具local7。这个工具是系统产生的错误的一部分。我们可以通过提供syslog:facility作为一个参数来指定一个工具。syslog工具的例子有mail、uucp、local0、local1等。要获取完整的列表，请查看系统所包含的syslog的文档（在命令行尝试man syslogd或man syslogd.conf）。下面是记录到syslog的一个例子。

```
ErrorLog syslog:local6
```

### 26.2.4 LogLevel指令

Apache所提供的错误信息有几种重要程度，我们可以选择只记录重要的信息，而忽略提示信息或无关紧要的警告信息。LogLevel指令接受一个错误级别参数。只有重要级别或更高级别的错误才会记录。

表26-2为LogLevel指令指定了有效值，就像Apache文档中所指定的一样。默认情况下，LogLevel的值是warn。对于大多数Apache安装来说，这应该足够了。如果你要尝试对一个特定配置进行故障排除，可以把级别修改为debug。

表26-2 Apache文档中描述的LogLevel选项

设置	说 明	示 例
emerg	Emergencies（紧急） ——系统不可用 ^	Child cannot open lock file. Exiting

alert	必须立即采取措施	getpwuid: couldn't determine user name from uid
crit	重要条件	socket: Failed to get a socket, exiting child
error	错误条件	Premature end of script headers
warn	警告条件	Child process 1234 did not exit, sending another
		SIGHUP
notice	一般但重要条件	httpd: caught SIGBUS, attempting to dump core in ^
info	信息	Server seems busy, (You may need to increase StartServers, or Min/MaxSpareServers)
debug	调试级消息	Opening config file...

## 26.3 管理Apache日志

Apache提供了几种工具来管理日志。其他的Apache专用的第三方工具也可以使用，并且这里也提到了。由于Apache可以以CLF记录请求，大部分通用日志处理工具也可以用于Apache。

### 26.3.1 解析主机名

在本章前面，我们学习了如何使用HostNameLookups指令在做出请求的时候打开或关闭主机名解析。如果HostNameLookups设置为off（默认值），日志文件将只包含IP地址。随后，我们可以在UNIX上使用命令行logresolve工具或者在Windows上使用logresolve.exe来处理日志文件，并且把IP地址转换为主机名。

logresolve工具从标准输入读取日志条目，并且把结果输出到其标准输出。要读取一个文件或写入到一个文件，我们在UNIX和Windows上都可以使用重定向，如下所示。

```
logresolve < access.log > resolved.log
```

日志解析工具效率很高，因为它们在响应客户机请求的时候可以缓存结果并且不会引起任何延迟。

### 26.3.2 日志备份

在高流量的Web站点中，记录访问的日志文件可能很快就变得很大。我们应该有一种方法来定期备份日志，按照定义的时间间隔来保存并压缩旧的日志文件。



在Apache运行的时候，日志文件不应该被删除，因为服务器直接写入到它们。解决方案是使用一个中间程序来记录请求。反过来，这个程序将负责日志的备份。

Apache提供了UNIX上的rotatelog程序 and Windows上的rotatelog.exe程序来完成此任务。它接受3个参数：一个文件名，一个以秒为单位的备份周期，以及一个可选的、相对UTC(Coordinated Universal Time, 协调世界时)的分钟偏移量，示例如下。

```
TransferLog "|bin/rotatelog /var/logs/apachelog 86400"
```

每天都创建了一个新的日志文件，并且把当前日志移动到/var/logs（在命令的末尾，86400就是一天中的秒数）。

**提示：**

如果程序的路径中包含空格，我们可能需要通过在它们前面加上一个\\（反斜杠）来转义，例如，My\\Documents。这在Windows平台上尤其常见。

如果文件名包含%前缀选项，名字将作为strftime函数的输入处理，该函数把%选项转换为时间值。rotatelog工具的手册页面包含了一个完整的选项列表，下面是一个例子。

```
TransferLog "|bin/rotatelog /var/logs/apachelog%m_%d_%y 86400"
```

这个命令把当前月份、日期和年份添加到日志文件名中。

如果这个名字没有包含任何%格式的选项，以秒为单位的当前时间将会添加到备份文件的名称中。

### 26.3.3 日志分析

不管是有单个的服务器和日志文件，还是有一个服务器集群产生它们的日志文件，我们收集了日志之后，就可以分析它们并获取有关流量和访问者行为的信息。

有很多商业软件、共享软件以及自由软件应用程序可以用来进行日志分析和报告。两款最流行的开源应用程序是Webalizer(<http://www.mrunix.net/webalizer/>)和awstats (<http://awstats.sourceforge.net/>)。Wusage是一款物美价廉的商业软件，可以在<http://www.boutell.com/wusage/> 找到。

### 26.3.4 监视错误日志

如果我们在UNIX系统上运行Apache，可以使用tail命令行工具来监视实时的日志条目，不管是访问日志还是错误日志都可以，语法如下。

```
tail -f logname
```

其中，logname是Apache日志文件的路径。它将在屏幕上显示日志文件的最后几行，并且当条目添加到文件中的时候继续显示它们。

我们可以找到其他的程序，使我们通过扫描错误日志文件，找到指定错误、错误请求等，来识别问题并报告它们。ScanErrLog就是这样的一个程序，可以在<http://www.librelogiciel.com/software/> 找到它。

## 26.4 把自定义信息记录到一个数据库

在MySQL中创建自己的日志表，配合PHP代码片断，就可以帮助我们捕获对站点的具体页面的访问相关的信息。使用这些信息，我们可以创建自定义报表。这种方法比困难读取Apache日志文件更简单一些，尤其是当我们要查找访问信息的一个子集的时候。下面各节介绍了这一过程的一个简单版本。

### 26.4.1 创建数据库表

自定义日志方法的第一步是创建数据库表。如下的表创建命令在MySQL数据库中创建了一个名为access\_tracker的表，其中有ID、页面标题、用户代理和访问日期的字段。

```
CREATE TABLE access_tracker (  
    id INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    page_title VARCHAR(50),  
    user_agent TEXT,  
    date_accessed DATE  
);
```

接下来，我们可以创建向这个表写入内容的代码段。

### 26.4.2 创建PHP代码段

按照我们已有经验，代码段基本上只需要一小段代码。换句话说，代码段只执行简单的任务，因而并不是一个长脚本。在这个例子中，程序清单26.1中的代码段把一些基本信息写入到access\_tracker表。

程序清单26.1 访问记录的代码段

```

1: <?php
2: //set up static variables
3: $page_title = "sample page A";
4: $user_agent = getenv('HTTP_USER_AGENT');
5:
6: //connect to server and select database
7: $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB")
8:     or die(mysqli_error());
9:
10: //create and issue query
11: $sql = "INSERT INTO access_tracker (page_title,user_agent,date_accessed)
12:     VALUES ('$page_title', '$user_agent', now())";
13: $result = mysqli_query($mysqli, $sql) or die(mysqli_error($mysqli));
14:
15: //close connection to MySQL
16: mysqli_close($mysqli);
17: ?>

```

这段代码的用法很简单：将它放置在想要记录的每个页面的开始处。对于每个页面，把代码段中的\$page\_title的值修改为页面的实际标题。

现在，创建名为sample1.php的一个示例脚本，其中包含了程序清单26.1的内容，如下是程序清单26.2的其他内容。

程序清单26.2 示例HTML页面

```

1: <!DOCTYPE html>
2: <html>
3: <head>
4: <title>Sample Page A</title>
5: </head>
6: <body>
7: <h1>Sample Page A</h1>
8: <p>blah blah blah.</p>
9: </body>
10: </html>

```

创建这个文件的数个副本，使用不同的文件名和不同的\$page\_title值。然后，使用Web浏览器访问这些不同的页面，以填充日志表。

### 26.4.3 创建示例报表

当`access_tracker`表拥有了数据，可以创建一个简单的报表页面来发布这些信息。程序清单26.3中的代码创建了一个报表，它执行查询来计算结果以及使用的浏览器的概要。这些代码段将在程序清单之后说明。

程序清单26.3 创建一个访问报表

```

1:  <?php
2:  //connect to server and select database
3:  $mysqli = mysqli_connect("localhost", "joeuser", "somepass", "testDB")
4:      or die(mysqli_error());
5:
6:  //issue query and select results for counts
7:  $count_sql = "SELECT count(page_title) AS p_count FROM access_tracker";
8:  $count_res = mysqli_query($mysqli, $count_sql) or
    die(mysqli_error($mysqli));
9:
10: while ($count_info = mysqli_fetch_array($count_res)) {
11:     $all_count = $count_info['p_count'];
12: }
13:
14: //issue query and select results for user agents
15: $user_agent_sql = "SELECT DISTINCT user_agent, count(user_agent) AS
16:     ua_count FROM access_tracker GROUP BY user_agent
17:     ORDER BY ua_count desc";
18: $user_agent_res = mysqli_query($mysqli, $user_agent_sql)
19:     or die(mysqli_error($mysqli));
20:
21: //start user agent display block
22: $user_agent_block = "<ul>";
23:
24: //loop through user agent results
25: while ($row_ua = mysqli_fetch_array($user_agent_res)) {
26:     $user_agent = $row_ua['user_agent'];
27:     $user_agent_count = $row_ua['ua_count'];
28:     $user_agent_block .= "
29:     <li>".$user_agent."
30:         <ul>
31:             <li><em>accesses per browser: ".$user_agent_count."</em>
32:             </ul>
33:     </li>";
34: }
35:
36: //finish up the user agent block
37: $user_agent_block .= "</ul>";
38:
39: //issue query and select results for pages
40: $page_title_sql = "SELECT DISTINCT page_title, count(page_title) AS

```

```

41:             pt_count FROM access_tracker GROUP BY page_title
42:             ORDER BY pt_count desc";
43: $page_title_res = mysqli_query($mysqli, $page_title_sql)
44:             or die(mysqli_error($mysqli));
45:
46: //start page title display block
47: $page_title_block = "<ul>";
48:
49: //loop through results
50: while ($row_pt = mysqli_fetch_array($page_title_res)) {
51:     $page_title = $row_pt['page_title'];
52:     $page_count = $row_pt['pt_count'];
53:     $page_title_block .= "
54:     <li>".$page_title."
55:         <ul>
56:             <li><em>accesses per page: ".$page_count."</em>
57:         </ul>
58:     </li>";
59: }
60:
61: //finish up the page title block
62: $page_title_block .= "</ul>";
63:
64: //close connection to MySQL
65: mysqli_close($mysqli);
66: ?>
67: <!DOCTYPE html>
68: <html>
69: <head>
70: <title>Access Report</title>
71: </head>
72: <body>
73: <h1>Access Report</h1>
74: <p><strong>Total Accesses Tracked:</strong>
75: <?php echo "$all_count"; ?></p>
76: <p><strong>Web Browsers Used:</strong>
77: <?php echo "$user_agent_block"; ?></p>
78: <p><strong>Individual Pages:</strong>
79: <?php echo "$page_title_block"; ?></p>
80: </body>
81: </html>

```

第3行连接到数据库，以便我们可以根据access\_tracker表来执行查询。第7行到第8行执行查询来统计页面的总数，第15行到第19行计算用户代理访问的次数。第22行开始一个无序列表语句块，用于用户代理查

询的结果，而第25行到第34行遍历结果并创建列表，这个列表块在第37行结束。

第40行到第44行创建并执行一个查询来统计单独的页面。第47行开始一个无序列表语句块，用于显示这一查询的结果，第50行到第59行遍历这个结果并创建访问过页面的列表，这个无序列表块在第62行结束。

把这些代码放入到一个名为accessreport.php的文本文件中，然后将其放入到Web服务器文档根目录下。当访问这个报表时，将会看到如图26-1所示的结果，页面名称、计数和浏览器信息可能有所不同，但情况大致是这样。

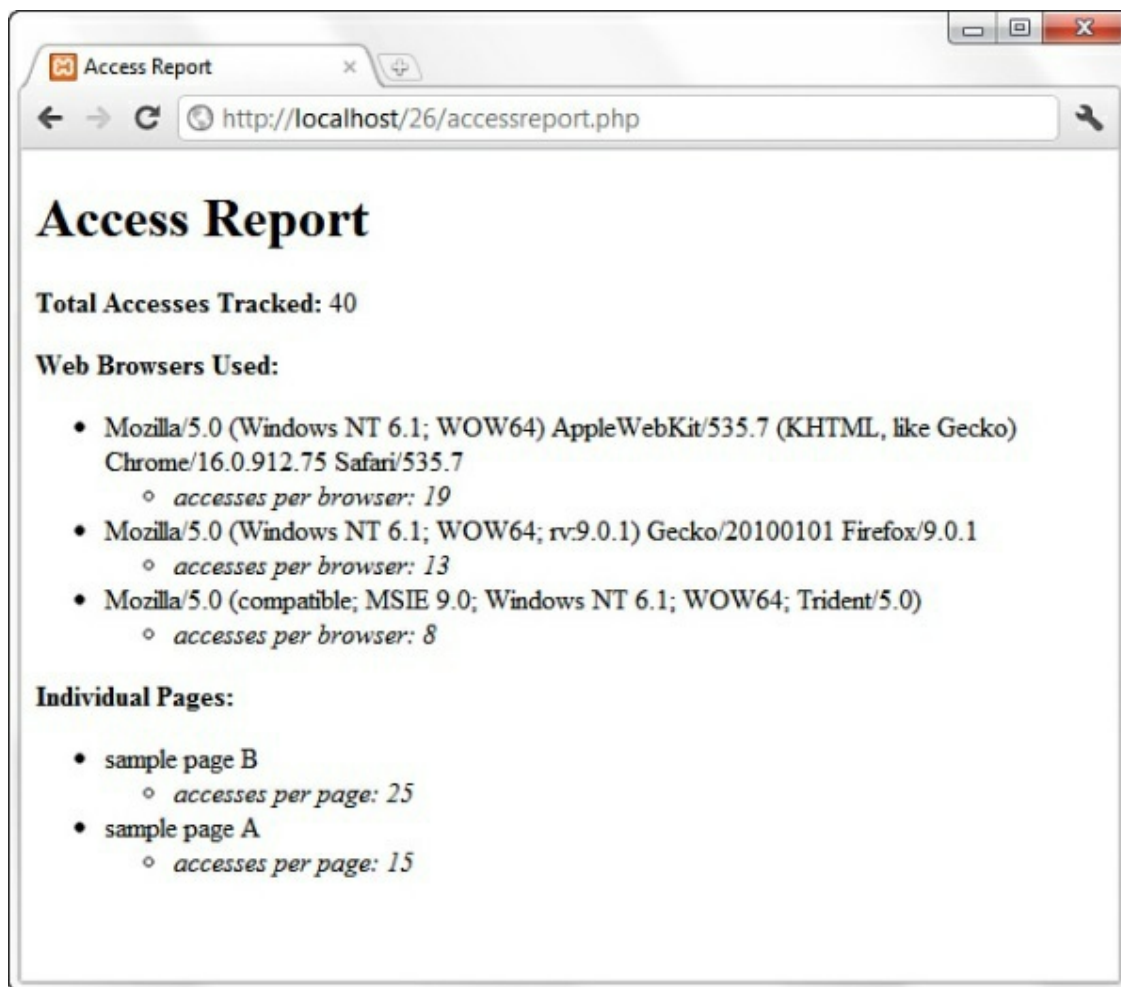




图26-1 记录页面的自定义访问报表

这种记录比辛苦读取Apache访问日志要容易很多，但是我们不建议使用数据库驱动的系统来完全替代访问日志。即便MySQL在系统上能够工作得很好，数据库连接也需要很大的开销。相反，把你的页面记录定为目标则特别重要。

## 26.5 小结

本章介绍了如何记录Apache所产生的有关请求和错误的特定信息。我们可以把日志存储到文件或数据库中，或者将它们传递给外部程序。我们学习了可以用来管理、处理和分析日志的不同工具，既有那些包含在Apache中的工具，也有那些来自第三方的可用工具。

最后，我们看到了一种简单的方法，它使用PHP代码段和一个MySQL数据库来执行特定页面的简单访问记录。这些信息随后以一个用PHP生成的、简单的访问报表的形式显示出来。

## 26.6 Q&A

**Q:** 为什么不希望记录图像？

**A:** 在负载较重的服务器上，日志可能成为一个瓶颈。如果日志的目的是为了记录访客的数量以及分析他们对网站的使用，我们可以通过只记录HTML页面来得到结果，而不用把图像包含在其中。这会减少存储在日志中的信息的数目，并且减少将其写入日志的时间。

## 26.7 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 如何避免从一个特定网络访问你的Web站点的客户端的信息被记入日志？
2. 如何把图像记录为一个不同的文件？
3. 为什么希望在你的Apache配置中关闭HostNameLookupoff？

## 解答

1. 在某些情况下，我们可能希望忽略来自某个特定网络的请求，例如自己的请求，以便它们不会改变结果。我们可以通过后处理日志删除它们，或者使用SetEnvIf指令来做到，如下所示。

SetEnvIf directive:

```
SetEnvIf Remote_Addr 10\.0\.0\. intranet
CustomLog logs/access_log "%h %l %u %t \"%r\" %>s %b" !intranet
```

2. 在本章前面，我们学习了如何避免记录图像。采用同样的环境变量的方法，我们可以把图像记录到另一个文件，而不是完全忽略图像，示例如下。

```
SetEnvIf Request_URI "(\\.gif|\\.jpeg)$" image
CustomLog logs/access_log common env=!image
CustomLog logs/images_log common env=image
```

3. 打开HostNameLookupson会导致服务器上额外的负担，因为它查找访问站点的用户的IP并将其写入到日志文件。稍后的日子，在产生使用报告的时候，你仍然可以使用主机名解析程序来获得信息，并由此将针对用户的服务器负载最小化。

## 思考题

1. 创建一个记录脚本，它记录了通过PHP对一个数据库的访问。使用可能的环境变量的列表，记录页面标题、用户代理和访问日期以外更多的信息。
2. 创建你所存储的访问数据的一个报表，添加一个日期范围选择器，并且只生成处于选定日期范围之内的结果。

## 第27章 应用程序本地化

在本章中，我们将学习：

- 如何识别不同的字符集并为此做好准备。
- 如何准备应用程序的结构并生成本地化站点。

World Wide Web中的关键词就是World Wide（意为世界范围）。使用PHP和MySQL创建一个可供讲不同语言的人使用的Web站点，这是小菜一碟。准备让你的应用程序在多种地方使用的过程，叫做国际化；为每一个地方而自定义代码的过程叫做本地化。



## 27.1 关于国际化和本地化

首先，不管是国际化还是本地化，都是与翻译类似的同一件事情。实际上，我们可以完全地翻译一个Web站点，全部用德语、全部用日语，或者全部用我们想要的任何一种语言，并且它不是考虑为一个国际化或本地化的Web站点。它只是一个翻译后的站点。国际化的关键部分如下所示。

- 使所有的字符串、图标和图形外部化。
- 修改格式化函数（日期、货币、数字等）的显示。

我们构建了自己的应用程序以使得字符串外部化（即函数、类和其他脚本中使用的所有字符串在一个地方管理，并且作为常量来导入或引用），并且格式化函数可以随着本地设置而变化之后，就开始了本地化的过程。翻译正是这个过程的一部分。

一个本地设置实际上是一个组，在这个例子中，是一组翻译后的字符串、图形、文本以及格式化惯例，它们将用于那些要本地化的应用程序或Web站点。这些组通常是用应用程序所使用的语言的名字来引用，例如German locale。尽管很明显German locale包含了翻译为德语的文本，但并不意味着这个站点只能供德国人使用，说德语的奥地利人也可以使用一个本地化的德语站点，但是，并没有称其为Austrian locale。

在下面的几节中，我们学习了如何使用不同字符集以及如何修改环境，以成功地做好应用程序本地化的准备。

## 27.2 关于字符集

字符集通常根据一种语言中用来定义一个字符所需的字节的数目，分为单字节字符集和多字节字符集。英语、德语和法语等还有很多其他的语言都是单字节语言，表示像字母a或数字9这样的一个字符，只需要一个字节。单字节代码集最多有256个字符，包括ASCII字符的完整集合、重音字符和其他所需的格式字符。

多字节代码集拥有多于256个的字符，包含所有的单字节字符作为其子集。多字节语言包括繁体中文和简体中文、日文、韩文、泰文、阿拉伯语、希伯来语等等，这些语言都需要多于一个字节来表示一个字符。一个好的例子就是单词Tokyo，即日本的首都东京。在英语中，它拼写为5个不同的字母，一共使用5个字节。然而，在日语中，这个单词用两个音节表示，分别是tou和kyou，每个音节使用两个字节，一共用到4个字节。

这就是字符集及其背后的技术的一个简要介绍，但是，实用性在于：为了以原本的语言正确地解释和显示Web页面中的文本，告诉Web浏览器使用哪一种字符集，这取决于你。这通过在发送所有的内容之前先发送相应的标头来做到。

如果你有一组包含日语文本的页面，并且没有发送关于语言和字符集的正确标头，在那些主语言不是日语的Web浏览器中将错误地显示这些页面。换句话说，由于没有包含字符集信息，浏览器假设用自己的默认字符集来显示文本。例如，如果日语页面使用Shift\_JIS或UTF-8字符

集，并且浏览器设置为ISO-8859-1，你的浏览器将会尝试使用单字节ISO-8859-1字符集来显示日语文本。这将会非常糟糕，除非标头提醒它使用Shift\_JIS或UTF-8，并且你的操作系统上安装了相应的库和语言包。

Mojibake是表示这种不能识别的字符的一个术语。要了解更多信息，参见<http://en.wikipedia.org/wiki/Mojibake>。所涉及的标头就是Content-type和Content-language标头，它们也设置为META标记。由于我们拥有了一个动态环境的所有工具，最好在文本之前发送相应的标头，并且在文档中显示正确的META标记。下面是header()函数的一个例子，它输出了一个英文站点的正确的字符信息。

```
header("Content-Type: text/html;charset=ISO-8859-1");
header("Content-Language: en");
```

相应的HTML5标签如下所示。

```
<html lang="en">
<meta charset="ISO-8859-1">
```

一个德语站点将会使用相同的字符集，但语言代码不同。

```
header("Content-Type: text/html;charset=ISO-8859-1");
header("Content-Language: de");
```

相应的HTML5标签如下所示。

```
<html lang="de">
<meta charset="ISO-8859-1">
```

一个日语站点使用不同的字符集和不同的语言代码。

```
header("Content-Type: text/html;charset=Shift_JIS");
header("Content-Language: ja");
```

相应的HTML5标签如下所示。

```
<html lang="ja">  
<meta charset="Shift_JIS">
```

## 27.3 环境修改

在本书的安装章节中所定义的环境，需要进行修改以处理本地化站点。尽管我们可以在Apache、PHP和MySQL中使用多个和语言相关的设置，以满足本地化站点的需求，但我们也可以不对配置做任何和语言相关的改变，从而执行本章中的所有任务。针对你自己的信息，下面各节将介绍为国际化而使用Apache、PHP和MySQL的相应文档。

### 27.3.1 Apache的配置修改

在第29章，我们将学习使用`mod_mime`或`mod_negotiation`模块以及`AddLanguage`和`AddCharset`指令（还有其他指令）来进行内容协商的概念。当我们手动改变了文件的扩展名，并且希望Apache根据这一扩展名来解释要使用的字符集的时候，需要用到这些指令。然而，这并非本章讨论的话题。我们希望你的所有本地化站点都具有相同的文件命名惯例（例如，`index.html`和`company_info.html`），而不是必须手动地创建具有基于不同语言的扩展名的多个页面来提供翻译的文件。我们对于站点本地化的目标是，在一个Web服务器上有一组使用正确翻译的文本填充的页面在运行。

#### 提示：

基于Apache的内容协商使用具有基于语言的命名惯例的多个文件并没有错。只不过这不是本章关注的重点。我们可以在<http://httpd.apache.org/docs.2.0/content.negotiation.html> 阅读到有关基于Apache的内容协商的更多内容。

### 27.3.2 PHP的配置修改

和Apache一样，对于本章的任何任务，也不需要PHP中做出配置改变。然而，如果需要的话，我们可以使用很多和处理多字节字符相关的函数。这些函数可以在位于<http://www.php.net/mbstring>的PHP手册中找到，并且必须在配置过程中使用如下代码使其变为可用（Windows用户在php.ini中支持php\_mbstring.dll扩展）。

```
--enable-mbstring=LANG
```

这里，LANG就是语言代码，例如，ja表示日语，cn表示简体中文等等。或者，我们可以使用这一行代码来支持所有可用语言。

```
--enable-mbstring=all
```

当mbstring函数在PHP中可用的时候，我们可以在php.ini配置文件中设置几个选项，以便正确地使用这些函数。在这些配置之后，我们可以使用超过40个和mbstring相关的函数中的任何一个来处理PHP中的多字节输入。

这些函数的手册非常全面，是进行多字节字符集和动态内容高级工作的推荐读物。尽管当你为了自学而阅读PHP手册的时候，我们推荐阅读这些内容，但是即便没有阅读它们，仅通过本章的内容，你已经了解得很好了。

### 27.3.3 MySQL的配置修改

本章中使用的本地化例子并不需要在MySQL中进行显式修改，因为这些例子不是数据库驱动的。MySQL中默认使用的字符集是ISO-8859-1，但是，这并不意味着我们仅限于在数据库中存储单字节字符。要了解和当前语言相关的MySQL元素，请阅读位于

<http://www.mysql.com/doc/en/Localisation.html> 的MySQL手册条目。

## 27.4 创建一个本地化页面结构

在本节中，我们将看到一个功能完备的例子。这是一个本地化的欢迎页面，它使用PHP使得用户可以选择一种目标语言并随后接收相应的文本。本节的目标是展示一个例子，将这个脚本中用到的字符串外部化，这是国际化的特征之一。

在这个脚本中，用户恰好在基于英语的Web站点上，但是也提供了在用户的本地进行浏览的选项，包括英语、德语和日语。这个过程中涉及如下3个元素。

- 创建并使用一个主文件来发送本地相关的标头信息。
- 创建并使用一个主文件来显示基于所选择的本地信息。
- 使用脚本本身。

程序清单27.1给出了用来发送本地相关的标头信息的主文件的内容。

程序清单27.1 语言定义文件

```
1:  <?php
2:  if ((!isset($_SESSION['lang'])) || (!isset($_GET['lang']))) {
3:      $_SESSION['lang'] = "en";
4:      $currLang = "en";
5:  } else {
6:      $currLang = $_GET['lang'];
7:      $_SESSION['lang'] = $currLang;
```



```

8:  }
9:
10: switch($currLang) {
11:     case "en":
12:         define("CHARSET","ISO-8859-1");
13:         define("LANGCODE", "en");
14:         break;
15:
16:     case "de":
17:         define("CHARSET","ISO-8859-1");
18:         define("LANGCODE", "de");
19:         break;
20:
21:     case "ja":
22:         define("CHARSET","UTF-8");
23:         define("LANGCODE", "ja");
24:         break;
25:
26:     default:
27:         define("CHARSET","ISO-8859-1");
28:         define("LANGCODE", "en");
29:         break;
30: }
31:
32: header("Content-Type: text/html;charset=".CHARSET);
33: header("Content-Language: ".LANGCODE);
34: ?>

```

程序清单27.1的第2行到第8行设置了存储用户选择的语言选项所需的会话值。

#### 提示：

下面的段落中列出的define\_lang.php或lang\_strings.php文件中并没有使用session\_start()函数，因为这些文件是通过主文件中的include()函数包含进去的。我们稍后创建的主文件，调用了session\_start()函数，它将对这些包含文件有效。

如果不存在会话值，英语本地设置将会被使用。如果你的站点默认是一个德语站点，我们将修改这个文件以便默认地使用德语本地设置。这个脚本为下一个脚本做准备，后者包含一个输入选择机制，通过把

\$currLang的值设置为第6行输入的结果。

第10行开始的switch语句包含了几个case语句，设计用来把相应的值赋给常量CHARSET和LANGCODE。第32行到第33行，在动态创建和发送Content-type和Content-language标头的时候，第一次真正使用这些常量。

把这个文件保存为define\_lang.php并且将其放置在Web浏览器的文档根目录下。这个文件定义了将在下一个脚本中使用的两个常量，下一个脚本是真正的显示脚本。这两个常量是CHARSET和LANGCODE，分别对应每个本地的字符集和语言代码。在显示脚本中，这些常量用来创建关于字符集和语言代码的、正确的META标记。尽管标头在这个脚本中发送，确保它们是页面本身的一部分以有助于任何需要的表单输入，这是一个好办法。

程序清单27.2创建了一个函数，它只是存储将要供显示脚本使用的外部化字符串。这个例子使用了两个字符串：一个用来欢迎用户访问页面（WELCOME\_TXT），另一个用来介绍语言选择过程（CHOOSE\_TXT）。

程序清单27.2 字符串定义文件

```

1:  <?php
2:  function defineStrings() {
3:      switch($_SESSION['lang']) {
4:          case "en":
5:              define("WELCOME_TXT", "Welcome!");
6:              define("CHOOSE_TXT", "Choose Language");
7:              break;
8:
9:          case "de":
10:             define("WELCOME_TXT", "Willkommen!");
11:             define("CHOOSE_TXT", "Sprache auswählen");
12:             break;
13:
14:          case "ja":
15:             define("WELCOME_TXT", "[unprintable characters]");
16:             define("CHOOSE_TXT", "[unprintable characters]");
17:             break;
18:
19:          default:
20:             define("WELCOME_TXT", "Welcome!");
21:             define("CHOOSE_TXT", "Choose Language");
22:             break;
23:      }
24:  }
25:  ?>

```

使用随书光盘中包含的lang\_strings.php文件，以使用这里无法显示的实际的日语字符。把这个文件放入到Web浏览器的文档根目录下。这个文件定义了两个常量，WELCOME\_TXT和CHOOSE\_TXT，它们用于显示脚本。这些常量定义于名为defineStrings()的函数中，尽管我们可以很容易地把这个文件编写为函数结构之外的一个长长的switch语句。我直接把它放入到一个函数是为了避免组织复杂语句，也是为了在使用显示脚本的时候能够更容易说明。

最后，到了创建显示脚本的时候。别忘了，国际化的一个关键因素是外部化所有的字符串，以便只需要使用一个主文件。程序清单27.3就是这样的一个例子。

```

1:  <?php
2:  session_start();
3:  include 'define_lang.php';
4:  include 'lang_strings.php';
5:  defineStrings();
6:  ?>
7:  <!DOCTYPE html>
8:  <html lang="<?php echo LANGCODE; ?>">
9:  <head>
10:    <title><?php echo WELCOME_TXT; ?></title>
11:    <meta charset="<?php echo CHARSET; ?>" />
12:  <body>
13:    <h1 style="text-align: center;"><?php echo WELCOME_TXT; ?></h1>
14:    <p style="text-align: center; font-weight: bold;">
15:      <?php echo CHOOSE_TXT; ?><br/><br/>
16:      <a href="<?php echo $_SERVER['PHP_SELF']. "?lang=en"; ?>">
17:        </a>
18:      <a href="<?php echo $_SERVER['PHP_SELF']. "?lang=de"; ?>">
19:        </a>
20:      <a href="<?php echo $_SERVER['PHP_SELF']. "?lang=ja"; ?>">
21:        </a>
22:    </p>
23:  </body>
24: </html>

```

你会注意到，程序清单27.3是一个基本的模板，因为在 `define_lang.php` 或 `lang_strings.php` 文件中，所有和语言相关的元素都外部化了。所有这3个文件确实都根据所选择的或默认的本地设置显示了正确的结果。

第5行调用了 `defineStrings()` 函数，它随后使得两个常量的值变为可用，这些常量在第8行、第10行、第11行、第13行和第15行使用。第16行到第18行显示了表示英语、德语和日语本地的旗帜，这些旗帜是可以点击的。当用户点击了这些旗帜中的一个，该本地设置将变为新的、被选择的本地设置，并且所用的字符串将是新的本地设置所对应的字符串。这些链接包含了 `lang` 变量，该变量作为 `$_GET['lang']` 传递到脚本中。如果查看一下程序清单27.1的第6行，将会看到它如何用来修改和

用户偏好的本地相关的设置。

尽管这里使用一面旗帜来表示语言选项只是为了在开发过程中进行说明，但是，并不推荐这种做法，因为没有一种本质的图形能够表示语言。例如，使用英国的旗帜表示英语，使用德国的旗帜表示德语。英语在80多个不同的国家都是官方或主要语言，而德语至少在10个国家是官方语言，没有一个旗帜能够表示这么多的信息。

把这个文件保存为lang\_selector.php并将其放置到Web浏览器的文档根目录下。当第一次访问它的时候，将会看到如图27-1所示的结果。



图27-1 第一次浏览语言选择器

默认的是英语，直到选择另一种语言。相应的，Welcome和Choose Language文本也以英文显示。当用户点击德国旗帜，他会看到图27-2；当用户点击日本旗帜，他会看到图27-3。



图27-2 查看德语页面



图27-3 浏览日文页面

提供Web站点的本地化版本的公司和组织，往往会花很长的时间讨论如何表示本地选项，包括旗帜、国家的名字等等。这没有清晰的答案，但是，请记住，不赞成使用旗帜。如何显示语言选项，这确实是一

个商业决策，但是，如果你经过了将字符串、文本和图像外化的过程，并且创建了一个国际化的Web站点模板，以备进行本地化，那么，本地选项的格式已经不是那么重要的。

## 27.5 使用gettext()来本地化应用程序

前面的小节介绍了应用国际化和本地化的一个基本过程。更加高级的方法，可能是使用内建的PHP函数gettext()，它是通往GNU gettext包（一个API）的一扇门。

要了解关于GNU gettext的更多信息，请访问<http://www.gnu.org/software/gettext/gettext.htm>。

要使用gettext及其PHP相关的函数，需要将目录文件翻译为特定的格式。这些文件的一种常用的跨平台编辑器，是Poedit（参见<http://www.poedit.com/>）。一旦创建了一个转换目录模板（都是外部化的字符串），可以将该模板发送给你聘请的翻译人员，或者，使用Transifex（<https://www.transifex.net/>）或Get Localization（<http://www.getlocalization.com/>）这样的服务。有了完整的目录文件，可以将它们放到Web服务器文档根目录下的目录中，并且开始使用gettext函数的过程。

可以从PHP Manual的<http://www.php.net/gettext> 页面了解和gettext以及gettext相关的PHP函数的更多信息。其中一些信息如下。

- 使用putenv()对LC\_ALL environment变量进行本地化设置。
- 使用setlocale()为LC\_ALL设置一个值（参见<http://www.php.net/setlocale>）。
- 使用bindtextdomain()为一个给定的域设置翻译目录的位置（在这种情况下，域指的是识别应用程序的一个名字，而不是



www.mydomain.com这样的域名，参见

<http://www.php.net/Bindtextdomain> )。

- 使用textdomain()设置和gettext一起使用的默认域（参见<http://www.php.net/textdomain>）。
- 从这里开始，使用gettext("some string")或("some string")，来针对该字符串调用gettext翻译。因此，如果你有一个翻译目录，负责将“Welcome”翻译为德语字符串“Willkommen!”，并且所有的环境变量都设置为适用于德语，如下的代码将会输出“Willkommen!”。

```
echo _("Welcome");
```

一旦了解了应用程序国际化和本地化的基础知识，如果你打算开发一款应用，供使用多种语言的人来用，我建议你深入了解一下基于gettext的本地化框架和众多的翻译服务（除非你有很多的本地语言人员供你差遣，或者有很多的钱可以用于翻译服务）。

## 27.6 小结

在本章中，我们介绍了国际化和本地化的基本知识。我们学习了创建一个国际化站点的两个关键：所有字符串、文本和图形的外部化，以及数字、货币和日期格式的外部化。我们还了解了，不管是国际化还是本地化都并不等同于翻译文本，翻译只是本地化的一部分。

我们还学习了一些有关字符集的知识：字符集可以是单字节的或者多字节的。我们还了解了发送正确的语言相关标头以便Web浏览器可以正确解释和显示文本的重要性。

最后，我们创建了一个实际的例子，来展示如何存储一个本地相关的会话变量，从而为一个已有的模板确定和发送本地化字符串。这个模板可以供所有的本地使用，因为每个元素都是外部化的。另外，我们还学习了使用应用程序框架进行本地化的高级步骤（使用PHP的gettext函数）。

## 27.7 Q&A

**Q:** 在**PHP**中，如何进行数字、日期和货币的本地化？

**A:** 这方面有两个非常有用的函数，`number_format()`和`date()`，我们已经学习了`date()`函数。要在一个本地化环境中使用它，只要重新排列月份、日期和年份元素，以适应本地格式（例如，`MM-DD-YYYY`, `DD-MM-YYYY`等等）。`number_format()`函数用于数字和货币，它根据本地选择，用逗号、句点或空格来分组千位。请阅读位于[http://www.php.net/number\\_format](http://www.php.net/number_format) 的PHP手册来了解可能的用法。

## 27.8 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 英语是一个单字节语言还是多字节语言？日语呢？
2. 在一个本地化站点中，与字符编码相关的两个关键标头是什么？
3. 除了文本字符串，在国际化一个站点的时候还有其他哪些内容元素需要注意？

## 解答

1. 英语是单字节的，日语是双字节的。
2. 带有字符集指示器的Content-Language和Content-Type。
3. 日期、货币和数字的格式化，是在国际化过程中需要注意的其他的内容元素。

## 思考题

1. 使用Google Translate（或者你知道的其他工具），给你在本章中创建的语言定义和显示文件，添加一些其他语言的“Welcome!”消息。
2. 由于不推荐将旗帜的图形用于表示选择语言，请将本章示例文件中基于旗帜的语言选项更改为其他更为合适的选项。

## 第28章 使用XML

在本章中，你将学习：

- 如何创建一个基本的**XML**文档结构。
- 如何使用**DOM**函数在**PHP**中访问**XML**。
- 如何使用**SimpleXML**函数在**PHP**中访问**XML**。
- 如何使用**JSON**数据。

本章介绍如何通过**PHP**使用**XML**文档和**JSON**数据，但并不是全面的介绍，否则需要编写关于这一主题的单独的一整本书来介绍。然而，对于刚刚接触**XML**的人，或者对于通过**PHP**操作**XML**的新手，以及接收并使用**JSON**数据的新手，一些函数比其他的更容易使用。本章介绍两组这样的函数。



## 28.1 什么是XML

XML源自这种语言的全名，即可扩展标记语言（Extensible Markup Language）。尽管这种语言名字中有标记一词，但不要认为XML和HTML一样，因为，除了这两种语言都基于标记对之外，它们就没有什么相似性了。XML用来存储和交换标记对之间的数据，而HTML对于内容中包含了什么或者它们的组织结构关心甚少，它唯一的目的是在浏览器中显示这些内容。

### 28.1.1 基本XML文档结构

XML文档包含了两个主要元素：处理指令和正文。处理指令包含了XML声明语句，以及想要添加的任何处理指令和注释。

提示：

要了解XML文档的完整定义，请参考位于<http://www.w3.org/TR/REC-xml>的XML规范。

如下的代码段是一个有效的处理指令。

```
<?xml version="1.0" ?>
<!-- Sample XML document -->
```

处理指令之后就是内容结构。XML是一个层级结构，就像书一样，一本书有标题和章，每一章包含段落，以此类推。在一个XML文档中，只有一个根元素。继续用图书的例子，这个元素可能叫做Books，并且标记<Books></Books>包围了所有的其他信息。

```
<Books>
```

接下来，为文档添加后续的元素，也叫做孩子。继续使用图书的例子，我们需要一个主图书元素，随后是其中用于书名、作者和出版信息的元素。我们把这些子元素叫做Title、Author和PublishingInfo。但是，出版信息可能包含多个信息，我们需要出版社的名字、位置和出版年份。这不是问题，只需要在父元素中（恰好也是根元素的一个子元素）创建另外一组子元素。例如，<PublishingInfo>元素看上去如下所示。

```
<PublishingInfo>
  <PublisherName>Sams Publishing</PublisherName>
  <PublisherCity>Indianapolis</PublisherCity>
  <PublishedYear>2012</PublishedYear>
</PublishingInfo>
```

综合起来，具有一个条目的一个示例books.xml文档如下所示。

```
<?xml version="1.0" ?>
<!--Sample XML document -->
<Books>
  <Book>
    <Title>A Very Good Book</Title>
    <Author>Jane Doe</Author>
    <PublishingInfo>
      <PublisherName>Sams Publishing</PublisherName>
      <PublisherCity>Indianapolis</PublisherCity>
      <PublishedYear>2012</PublishedYear>
    </PublishingInfo>
  </Book>
</Books>
```

记住创建有效的XML文档的如下两个重要规则。

- XML区分大小写，因此，<Book>和<book>被看作是不同的元素。
- 所有的XML标记必须正确地结束，XML标记必须正确地嵌套，并且不允许重复标记。

向books.xml文件添加一些虚构的条目并且将其放置到Web服务器的

文档根目录下供稍后的例子使用。我们可以在本章后面给出的不同界面的例子中使用同一个XML文件。

### 28.1.2 何时应该使用XML和PHP

这个问题的简短（且不耐烦）的回答是，“你想要使用的任何时候”，但是，你可以想象，正式的回答要比这个复杂一点。在本节的前面，我注意到XML定义并携带内容。这是事实。但是，这在实际中看起来是什么样的呢？

本章中的示例使用XML来存储一个较小的图书书目。想象一个以专有数据库格式存储的较大的图书书目，但是这个数据库能够以XML格式输出数据。如果你需要拿到这个书目，但是又不想购买或使用存储它的专有数据库，XML就是解决的方法。数据的所有者可以将目录导入为XML格式，而你随后可以解析它并显示想要的内容，只要使用本章前面所描述的格式之一就可以了。

接下来，我们将学习使用两组不同的函数来解析XML文档，它们分别是DOM函数和SimpleXML函数，而它们都产生相同的结果（解析XML文档以提供数据，以便在随后的脚本中可以使用这些数据），这两种方法彼此略有不同。例如，尽管SimpleXML函数比DOM函数更易于使用，然而，正如你在随后的程序清单中看到的，当使用较大的文件，由于在开始使用文件之前，整个XML文档都会加载到内存中并进行解析，因此，将会带来性能的损失。选择哪种方法，需要有所权衡，这就是为什么本章只是介绍两组函数，而让你自己设法搞清楚它们实际上能为你和你的开发环境做些什么，以及不能做什么。

## 28.2 使用DOM函数在PHP中访问XML

自从PHP 4开始，DOM XML扩展就成为PHP的一部分，而在PHP 5中则是全面接受。主要改变在于，PHP的默认安装中包含了DOM功能，这就确保了要使用这些函数不需要安装和配置附加的库或扩展。

提示：

DOM表示文档对象模型（Document Object Model）。要了解DOM的更多信息，请参考<http://www.w3.org/TR/DOM-Level-2-Core/core.html>。

DOM函数的主要目的是允许我们使用DOM API来操作存储在一个XML文档中的数据。最基本的DOM函数是DOMDocument->load()，它根据一个文件的内容来创建一个新的DOM树。在创建了DOM树之后，我们可以使用其他的DOM函数来操作数据。在程序清单28.1中，DOM函数用来遍历DOM树，并获取存储的值以供稍后显示。

程序清单28.1 使用DOM函数遍历XML文档

---

```
1: <?php
2: $dom = new DomDocument;
3: $dom->load("books.xml");
4:
5: foreach ($dom->documentElement->childNodes as $books) {
6:     if (($books->nodeType == 1) && ($books->nodeName == "Book")) {
7:
8:         foreach ($books->childNodes as $theBook) {
9:             if (($theBook->nodeType == 1) &&
10:                ($theBook->nodeName == "Title")) {
11:                 $theBookTitle = $theBook->textContent;
12:             }
13:
14:             if (($theBook->nodeType == 1) &&
15:                ($theBook->nodeName == "Author")) {
16:                 $theBookAuthor = $theBook->textContent;
17:             }
18:
19:             if (($theBook->nodeType == 1) &&
20:                ($theBook->nodeName == "PublishingInfo")) {
21:
22:                 foreach ($theBook->childNodes as $thePublishingInfo) {
23:                     if (($thePublishingInfo->nodeType == 1) &&
24:                        ($thePublishingInfo->nodeName == "PublisherName")) {
25:                         $theBookPublisher = $thePublishingInfo->textContent;
26:                     }
27:
28:                     if (($thePublishingInfo->nodeType == 1) &&
29:                        ($thePublishingInfo->nodeName == "PublishedYear")) {
30:                         $theBookPublishedYear =
31:                             $thePublishingInfo->textContent;
32:                     }
33:                 }
34:             }
35:         }
36:
37:         echo "
38:         <p><em>".$theBookTitle."</em>
39:         by ".$theBookAuthor."<br/>
40:         published by ".$theBookPublisher." in ".$theBookPublishedYear."</p>";
41:
42:         unset($theBookTitle);
43:         unset($theBookAuthor);
44:         unset($theBookPublisher);
45:         unset($theBookPublishedYear);
46:     }
47: }
48: ?>
```

---

在第2行, 创建了一个新的DOM文档, 在第3行, books.xml的内容

载入到这个文档中。正如我们在随后的代码行中所见到的那样，这个文档树现在可以通过\$dom访问。第5行开始遍历文档树的主循环，它把文档中的每个节点都放入到一个名为\$books的数组中。

第6行查找一个名为“Book”的元素，并且如果找到就继续处理。别忘了，<Book>

</Book>标记对包含了books xml文件中一本书的所有条目。如果继续处理，第8行把所有子节点都收集到一个名为\$theBook的数组中，并且，第9行到第12行以及第14行到第17行的if语句分别查找名为“Title”和“Author”的特定节点，并且将其值放入到\$theBookTitle和\$theBookAuthor变量中以供稍后使用。

第19行开始一个类似的if语句，但是由于这一行查找一个名为“Publishing Info”的节点，并且你知道<PublishingInfo></PublishingInfo>标记对进一步包含了子节点集合，所以需要另一个循环结构来获取下一个数据层级中的信息。在第22行，找到了子节点并且将其放置到名为\$thePublishingInfo的数组中，随后，第23行到第26行以及第28行到第32行的if语句分别查找名为“PublisherName”和“PublishedYear”的特定节点，并且将它们的值放入到\$theBookPublisher和\$theBookPublishedYear变量中以供稍后使用。

第8行所创建的循环在第35行结束后，第37行到第40行使用存储在\$theBookTitle、\$theBookAuthor、\$theBookPublisher和\$theBookPublishedYear变量中的值向浏览器显示一个标记的字符串。使用了这些值之后，在第42行到第45行重置它们，并且循环查找文档树中下一个“Book”条目。

把这个程序清单保存为domexample.php并且将其放置在Web服务器的文档根目录下。当通过Web浏览器查看它的时候，应该看到图28-1所示的结果。

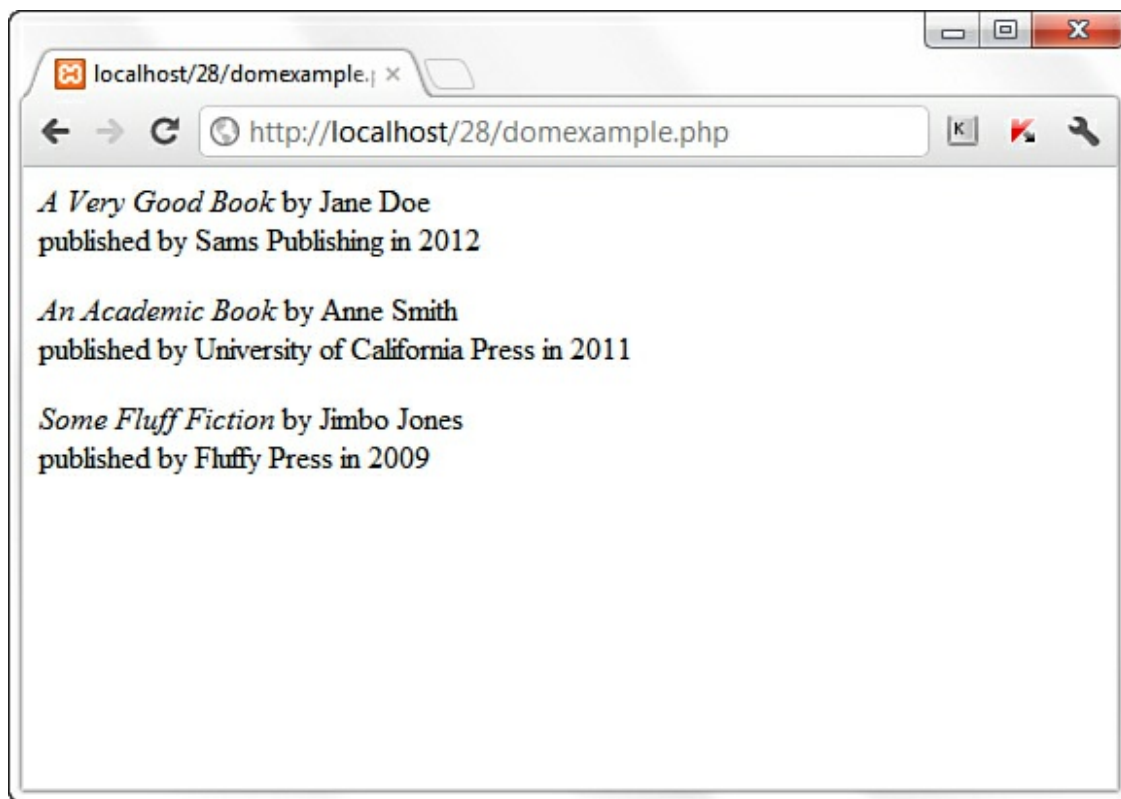


图28-1 使用DOM函数提取和显示的文本

要了解PHP中的80多个DOM相关函数的完整列表，包括那些添加到XML文档并保存新版本的函数，请参阅位于<http://www.php.net/dom>的PHP手册。

在下一节中，我们将使用同样的books.xml文件，但是将使用SimpleXML函数组来获取和显示其值。

## 28.3 使用SimpleXML函数在PHP中访问XML

SimpleXML是PHP 5中新增加的一个函数，它在默认情况下可以使用并且不需要任何额外的安装和配置步骤。它就像PHP手册中所描述的一样，是“转换XML的一个非常简单而且易于使用的工具集”，而且它的功能很强大。

和DOM函数组包含大量函数和方法不同，只有少数几个SimpleXML函数和方法，目前算来，只有7个。最基本的SimpleXML函数把XML数据转换为一个对象，可以直接访问和操作这个对象，而不需要特殊的函数来做到这一点。我们需要知道的第一个函数就是`simplexml_load_file()`，它载入一个文件并创建一个拥有数据的对象，如下所示。

```
$object_with_data = simplexml_load_file("somefile.xml");
```

在程序清单28.2中，短短几行代码用来创建一个SimpleXML对象，然后显示存储在对象中的层级数据。

程序清单28.2 使用SimpleXML载入并显示数据

---

```
1: <?php
2: $theData = simplexml_load_file("books.xml");
3: echo "<pre>";
4: print_r($theData);
5: echo "</pre>";
6: ?>
```

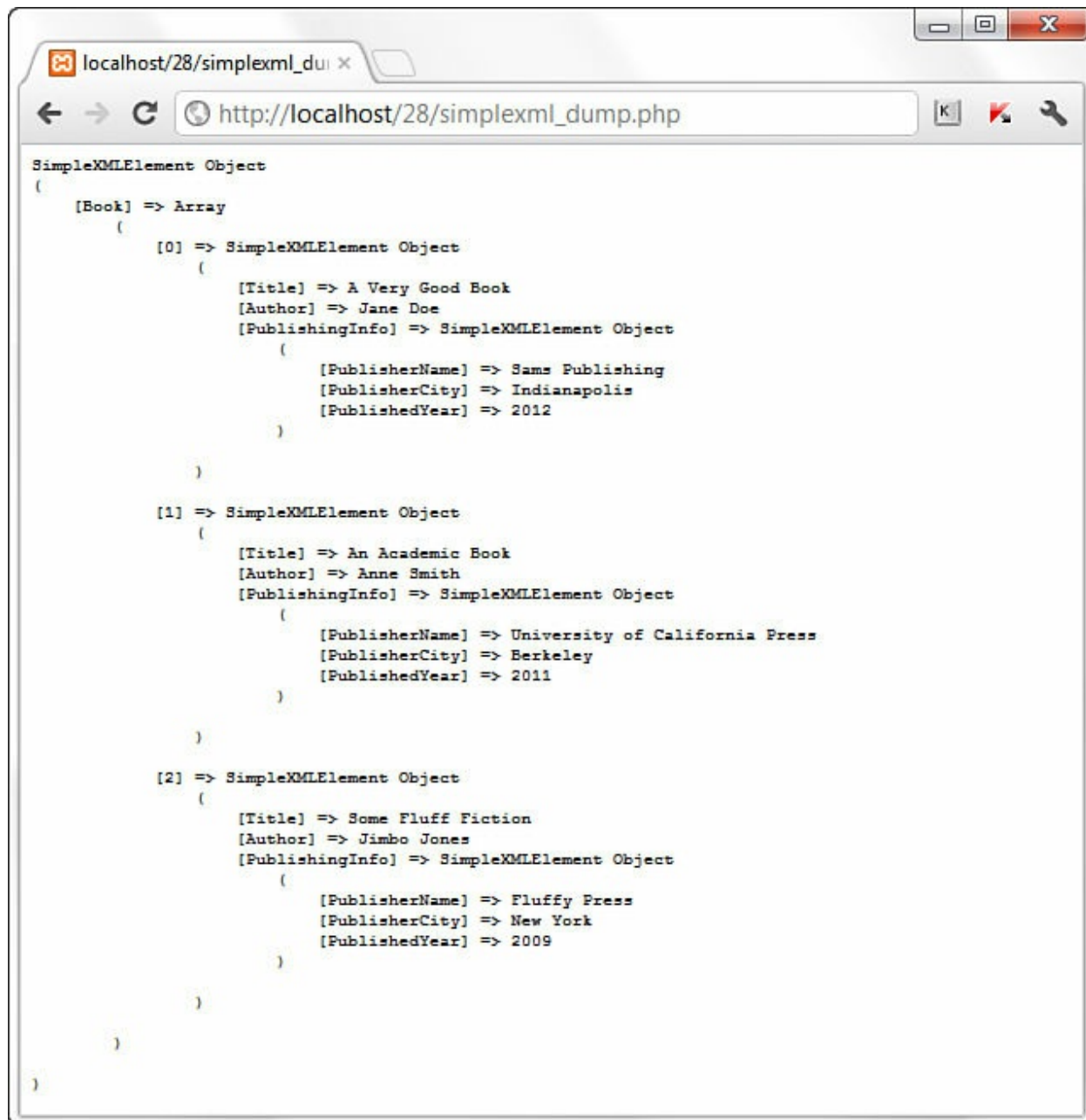
---

在第2行，使用`simple_load_file()`把books.xml的内容载入到一个名为`$theData`的对象中。在第4行，使用`print_r()`函数来输出存储在对象中的



数据的一个可读版本，这些数据用<pre></pre>标记对包围起来。

把这个程序清单保存为simplexml\_dump.php并且将其放置在Web服务器的文档根目录下。当通过Web浏览器查看它的时候，应该看到图28-2所示的结果。

A screenshot of a web browser window. The address bar shows 'http://localhost/28/simplexml\_dump.php'. The page content displays a PHP var\_dump of a SimpleXMLElement object. The object has a 'Book' property which is an array of three SimpleXMLElement objects. Each book object contains 'Title', 'Author', and 'PublishingInfo' properties. The 'PublishingInfo' property is itself a SimpleXMLElement object with 'PublisherName', 'PublisherCity', and 'PublishedYear' properties.

```
SimpleXMLElement Object
(
    [Book] => Array
        (
            [0] => SimpleXMLElement Object
                (
                    [Title] => A Very Good Book
                    [Author] => Jane Doe
                    [PublishingInfo] => SimpleXMLElement Object
                        (
                            [PublisherName] => Sams Publishing
                            [PublisherCity] => Indianapolis
                            [PublishedYear] => 2012
                        )
                )
            [1] => SimpleXMLElement Object
                (
                    [Title] => An Academic Book
                    [Author] => Anne Smith
                    [PublishingInfo] => SimpleXMLElement Object
                        (
                            [PublisherName] => University of California Press
                            [PublisherCity] => Berkeley
                            [PublishedYear] => 2011
                        )
                )
            [2] => SimpleXMLElement Object
                (
                    [Title] => Some Fluff Fiction
                    [Author] => Jimbo Jones
                    [PublishingInfo] => SimpleXMLElement Object
                        (
                            [PublisherName] => Fluffy Press
                            [PublisherCity] => New York
                            [PublishedYear] => 2009
                        )
                )
        )
)
```

图28-2 来自一个SimpleXML对象的数据

输出数据并不是都像这样壮观，但是，它确实展示了对象的结构，这反过来使我们知道了如何以层级的方式访问数据。例如，`simplexml_dump.php`的输出显示了一本书的条目。

```
[0] => SimpleXMLElement Object
(
    [Title] => A Very Good Book
    [Author] => Jane Doe
    [PublishingInfo] => SimpleXMLElement Object
    (
        [PublisherName] => Sams Publishing
        [PublisherCity] => Indianapolis
        [PublishedYear] => 2012
    )
)
```

要直接引用这条记录，用法如下。

```
$theData->Book
```

你可以在记录中访问元素，如下所示。

- `$theData->Book->Title` for the Title
- `$theData->Book->Author` for the Author
- `$theData->Book->PublishingInfo->PublisherName` for the Publisher Name
- `$theData->Book->PublishingInfo->PublisherCity` for the Publisher City
- `$theData->Book->PublishingInfo->PublishedYear` for the Published Year

但由于我们可能希望遍历所有记录而不只是一条记录，对数据的引用可能有些不同，这在程序清单28.3中可以看到。

---

```
1: <?php
2: $theData = simplexml_load_file("books.xml");
3:
4: foreach($theData->Book as $theBook) {
5:     $theBookTitle = $theBook->Title;
6:     $theBookAuthor = $theBook->Author;
7:     $theBookPublisher = $theBook->PublishingInfo->PublisherName;
8:     $theBookPublisherCity = $theBook->PublishingInfo->PublisherCity;
9:     $theBookPublishedYear = $theBook->PublishingInfo->PublishedYear;
10:
11:     echo "
12:     <p><em>".$theBookTitle."</em>
13:     by ".$theBookAuthor."<br/>
14:     published by ".$theBookPublisher." (".$theBookPublisherCity.")
15:     in ".$theBookPublishedYear."</p>";
16:
17:     unset($theBookTitle);
18:     unset($theBookAuthor);
19:     unset($theBookPublisher);
20:     unset($theBookPublishedYear);
21: }
22: ?>
```

---

在第2行，使用`simple_load_file()`把books.xml的内容载入到一个名为`$theData`的对象中。在第4行，把`$theData->Book`的内容（即所有单个记录）放入到了一个名为`$theBook`的数组中。第5行到第9行收集了具体元素的值，从`$theBook`的层级开始，并且这些值在第11行到第15行输出。第17行到第20行重置了变量的值以供下一次循环使用。

把这个程序清单保存为`simplexmlexample.php`并且将其放置在Web服务器的文档根目录下。当通过Web浏览器查看它的时候，应该看到图28-3所示的结果。



图28-3 使用SimpleXML函数提取和显示的文本

注意，输出看上去和程序清单28.1的输出很类似，并且，实际上，SimpleXML示例只是我们前面见到的基于DOM的示例的一个简单（或者说更加精炼的）版本。

要了解关于PHP中SimpleXML函数的更多信息，请参阅位于<http://www.php.net/simplexml> 的PHP手册。

## 28.4 使用JSON

JSON表示JavaScript Object Notation（JavaScript对象表示法），这是另一种数据交换格式（像XML一样），对于人和机器来说，它都是很容易阅读和编写的。由于其简单性，JSON输出变得越来越普及，并且可能有一天会通过API取代（如果还没有发生这种情况的话）XML用于数据显示的输出。

使用JSON，我们可以有名/值对的一个集合（它采用对象的形式），并且，可以有值的一个有序列表（它采用数组的形式）。如果我们将本章前面的books.xml文件中的条目重新写成JSON的格式，它看上去如下所示。

```
{
  "book":[
    {
      "title":"A Very Good Book",
      "author":"Jane Doe",
      "publisher_name":"Sams Publishing",
      "publisher_city":"Indianapolis",
      "publisher_year":"2012"
    }
  ]
}
```

添加两个其他的条目，将会得到一些JSON格式的数据，如程序清单28.4所示。

程序清单28.4 JSON格式的图书数据

---

```
1: {
2:   "book":[
3:     {
4:       "title":"A Very Good Book",
5:       "author":"Jane Doe",
6:       "publisher_name":"Sams Publishing",
7:       "publisher_city":"Indianapolis",
8:       "publisher_year":"2012"
9:     },
10:    {
11:      "title":"An Academic Book",
12:      "author":"Anne Smith",
13:      "publisher_name":"University of California Press",
14:      "publisher_city":"Berkeley",
15:      "publisher_year":"2011"
16:    },
17:    {
18:      "title":"Some Fluff Fiction",
19:      "author":"Jimbo Jones",
20:      "publisher_name":"Fluffy Press",
21:      "publisher_city":"New York",
22:      "publisher_year":"2009"
23:    }
24:  ]
25: }
```

---

要了解有关JSON的更多知识，参见<http://www.json.org>。在初次创建JSON的时候，一款有用的工具是JSON解析器，位于<http://json.parser.online.fr/>，它使你能够解析文本并找到（和修正）语法错误。

一旦有了JSON数据，你可以使用PHP的json\_decode()函数来接受所有格式良好的数据，并且将它们转换为一个对象，就像前面的SimpleXML示例中所做的那样。程序清单28.5使用了一段简短的代码来载入一些JSON数据，并且显示了对象中存储的数据的层级。

程序清单28.5 加载并显示JSON数据

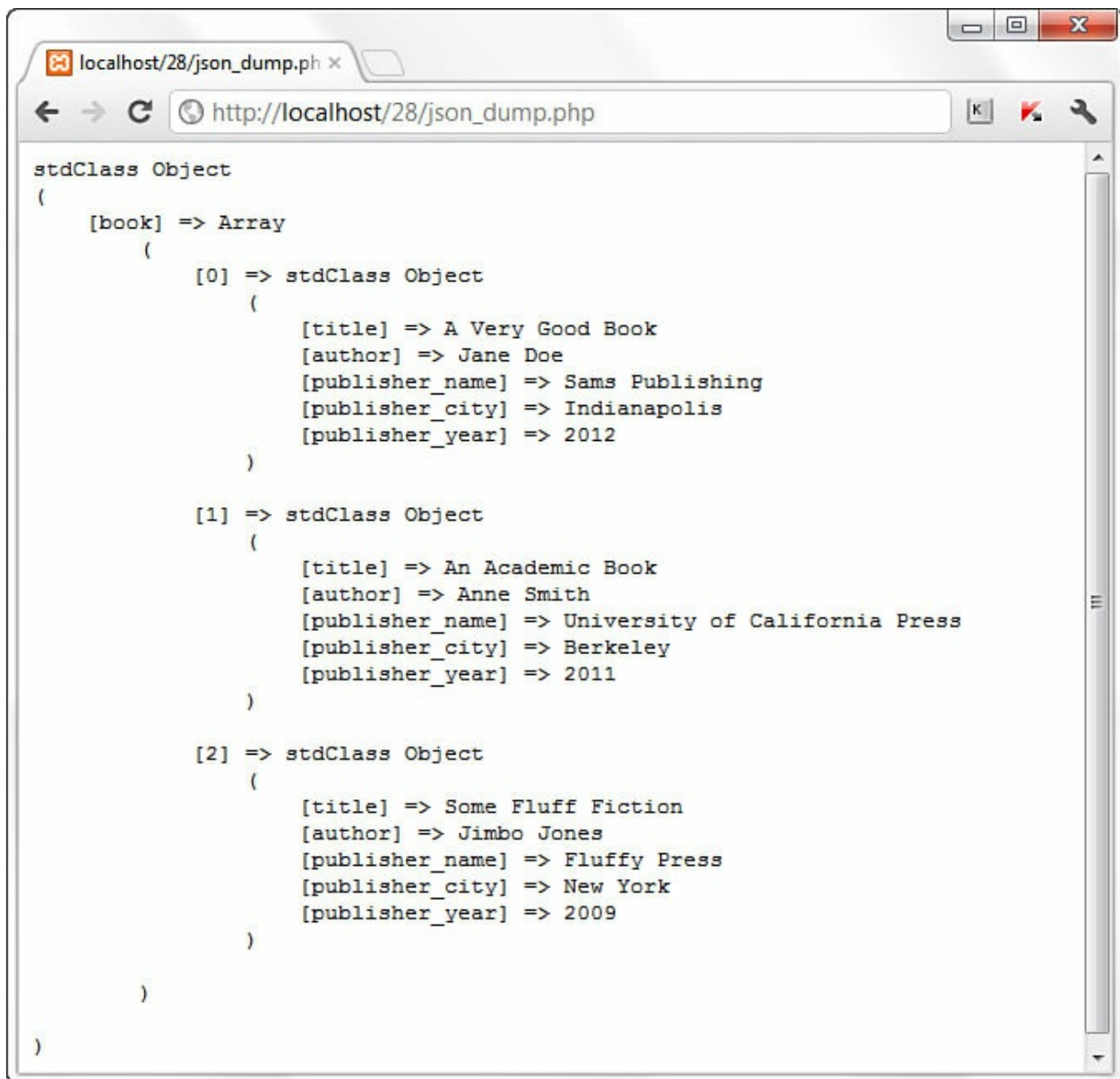
---

```
1: <?php
2: $theData = file_get_contents("books.txt");
3: echo "<pre>";
4: print_r(json_decode($theData));
5: echo "</pre>";
6: ?>
```

---

第2行使用`file_get_contents()`，将`books.txt`（包含了程序清单28.4中的JSON数据的一个文本文件）的内容载入到一个名为`$theData`的对象中。在第4行，`print_r()`函数输出了对象中所存储的JSON数据解码后的一个可读版本，其内容用`<pre></pre>`标签括了起来。

将这个列表保存为`json_dump.php`，并且将其放置到Web服务器的文档根目录下。通过Web浏览器来浏览的时候，应该会看到如图28-4所示的内容。



```
stdClass Object
(
    [book] => Array
        (
            [0] => stdClass Object
                (
                    [title] => A Very Good Book
                    [author] => Jane Doe
                    [publisher_name] => Sams Publishing
                    [publisher_city] => Indianapolis
                    [publisher_year] => 2012
                )

            [1] => stdClass Object
                (
                    [title] => An Academic Book
                    [author] => Anne Smith
                    [publisher_name] => University of California Press
                    [publisher_city] => Berkeley
                    [publisher_year] => 2011
                )

            [2] => stdClass Object
                (
                    [title] => Some Fluff Fiction
                    [author] => Jimbo Jones
                    [publisher_name] => Fluffy Press
                    [publisher_city] => New York
                    [publisher_year] => 2009
                )
        )
)
```

图28-4 数据的JSON格式

要创建这个数据的一个格式化版本，可以像下面这样访问记录中的元素。

- `$theData->book->title` for the Title
- `$theData->book->author` for the Author
- `$theData->book->publisher_name` for the Publisher Name



- `$theData->book->publisher_city` for the Publisher City
- `$theData->book->publisher_year` for the Published Year

正如前面所提到的，使用JSON的最常见的方式，就是作为API的输出。如下的URL包含了一个示例的API端点，用于访问Google搜索API，它带有两个变量：`v`表示API的版本（这里是1.0），`u`表示搜索关键词（这里是PHP）。

```
http://ajax.googleapis.com/ajax/services/search/web?v=1.0&q=PHP
```

将程序清单28.5的第2行更改为如下内容，将该文件保存为 `json_google_dump.php`，并将其放置到你的Web服务器的文档根目录下。

```
$theData = file_get_contents("http://ajax.googleapis.com/ajax/services/
search/web?v=1.0&q=PHP");
```

当你通过Web浏览器查看这段脚本的时候，会看到如图28-5所示的结果。

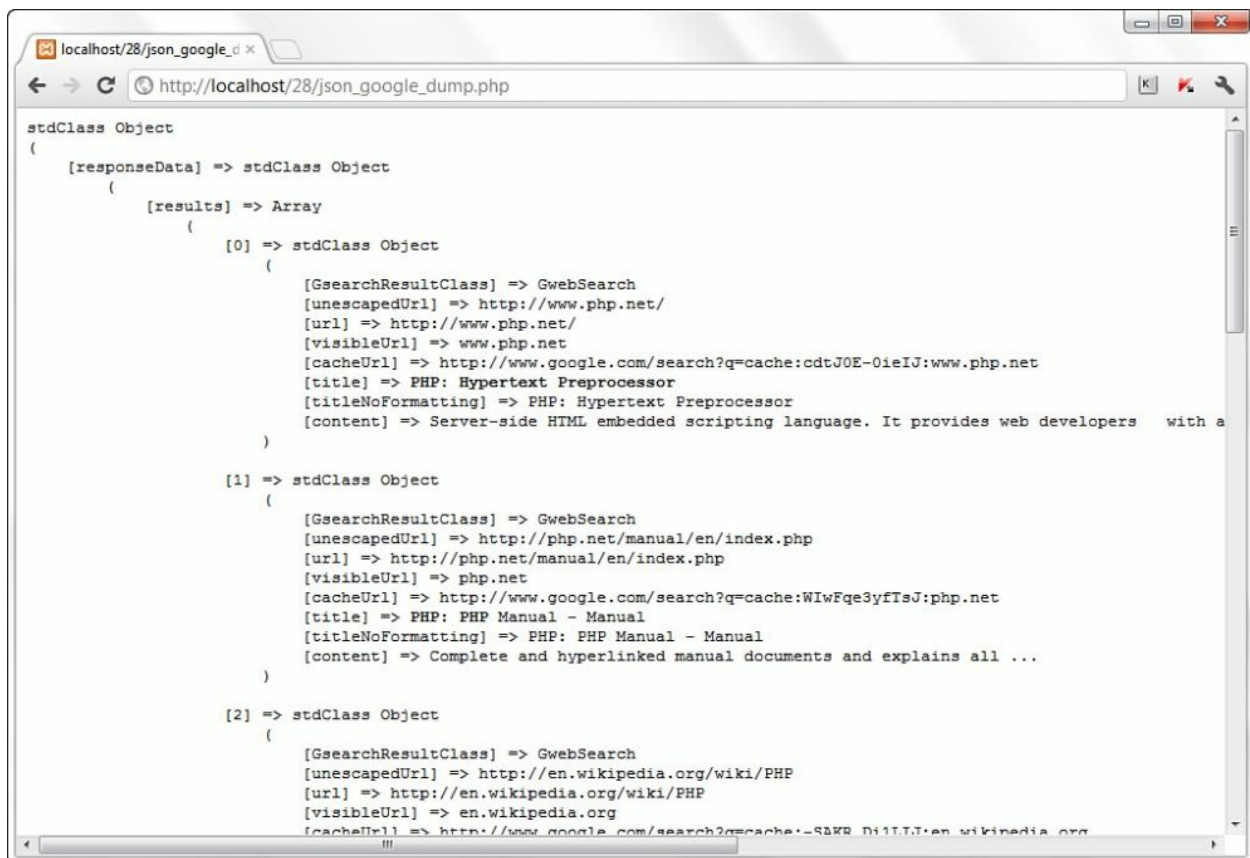


图28-5 一个Google搜索的JSON输出

一旦了解了使用JSON数据的基础知识（在使用了PHP中的 `json_decode()` 函数之后），那么，操作对象就变得很基础了，互联网上所有的数据都尽在你的掌握之中。哦，并不是所有的，但已经相当多了。要了解有关API的完整列表和更多信息，请访问 ProgrammableWeb，地址是<http://www.programmableweb.com/>。

## 28.5 小结

这个简短的一章介绍了两组用来操作XML（DOM函数和SimpleXML）和JSON的PHP函数。除了对这两个话题的简短概览，我们还给出了例子，分别使用这两组函数来显示存储在XML或JSON中的信息。本章的目的是介绍使用PHP操作XML和JSON的概念。如果你对于综合使用XML和PHP感兴趣，也可以看看AJAX(Asynchronous JavaScript and XML，异步JavaScript和XML)，它通常使用PHP来产生或修改XML或JSON数据，然后再将其显示给客户端。

## 28.6 Q&A

**Q:** 既然MySQL已经是一款很好的（而且免费的）数据库，为什么还要使用XML来存储数据呢？

**A:** XML不仅可以用作一种存储方法，而且可以用作数据传输的一个中介。例如，你可以使用XML来连接一个数据库，提取数据并将其发送给只能解释XML数据的第三方函数。此外，尽管MySQL确实是一款很好的（而且免费的）数据库，一些用户可能不能访问MySQL或者任何其他数据库，在这种情况下，XML文件可以充当一个数据库系统的角色。

**Q:** 如何使用脚本的其他部分所创建的数组和对象来创建JSON？

**A:** 如果想要生成JSON输出，你只需要使用json\_encode()函数，它接受已有的数组和对象并且将它们转换为JSON格式。参见[http://www.php.net/json\\_encode](http://www.php.net/json_encode) 了解详细信息。

## 28.7 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 一个有效的XML文档的开始行应该是什么？
2. 如下的代码把XML内容放入到一个新的DOM文档中了吗？

```
$dom = new XmlDocument;
```

3. 使用什么代码来把一个名为my.xml的文件的内容载入到一个名为\$myData的SimpleXML对象中？

## 解答

1. `<?xml version="1.0">`

2. 没有，它只是创建了一个引用为\$dom的DOM文档。要载入内容，必须还要使用如下代码。

```
$dom->load("books.xml");
```

3. `$myData = simplexml_load_File("my.xml");`

## 思考题

1. 创建一个脚本，将JSON编码的图书数据格式化，从而在浏览器中生成与程序清单28.1和程序清单28.3相同的结果。
2. 生成一个数据的数组，手动地或者通过脚本从数据库获取信息，然后，将其编码为JSON格式。如果随后创建另一个脚本来载入该输出并格式化它以备使用，那么，你就可以为自己的应用程序创建和使用一个API。



## 第6部分 管理和优化

第29章 **Apache**性能调校和虚拟主机

第30章 建立一个安全的**Web**服务器

第31章 优化和调校**MySQL**

第32章 软件升级

第33章 使用应用程序框架

## 第29章 Apache性能调校和虚拟主机

在本章中，你将学习：

- 操作系统以及**Apache**的哪些相关设置可能限制服务器的扩展性或降低性能。
- 有关载入测试**Apache**的几种工具。
- 如何对**Apache**调优。
- 如何配置**Apache**以检测和防止来自客户机的滥用。
- 如何配置基于名字的虚拟主机、基于**IP**的虚拟主机，以及二者之间的不同。
- 关于虚拟主机对**DNS**的依赖性。
- 如何设置大量的同样的虚拟主机。

本章和管理相关，焦点将是提高**Apache**服务器的性能和可扩展性。此外，我们还将学习基于名字和基于**IP**的虚拟主机，**DNS**相关的问题以及和**Web**浏览器自身相关的问题。本章还说明了可以用来把客户机相互隔离开的不同方法，以及相关的安全性取舍。

## 29.1 可扩展性问题

本节介绍可扩展性问题以及如何防止这些问题。本节更像是一个“不要这么做”的列表，说明了可能会降低性能或阻碍服务器升级的限制性因素。我们还探讨了Apache的预前调校以优化性能。

### 29.1.1 操作系统限制

有几个操作系统因素可能会妨碍Apache的升级。这些因素和进程创建、内存限制以及同时打开的文件或连接的最大数目等有关。

提示：

UNIX `ulimit`命令使我们可以每个进程的基础上来设置本节所介绍的几个限制。请参见操作系统文档以了解`ulimit`命令语法的详细内容。

#### 1. 进程

Apache提供了用来防止服务器进程和线程数目超过某个限制的设置。这些设置影响到可扩展性，因为它们限制了同时到Web服务器的连接的数目，这反过来影响到我们可以同时服务的访问者的数目。

Apache多进程模块（Multi-Processing Module, MPM）设置反过来可以通过OS设置限制进程和线程的数量。如何在不同的操作系统之间修改这些限制？在Linux内核中，需要修改`/proc/sys/kernel/threads-max`文件中定义的`NR_TASKS`。我们可以使用如下这条命令来读取文件的内容。

```
# cat /proc/sys/kernel/threads-max
```

我们可以使用如下这条命令写入文件。

```
# echo value > /proc/sys/kernel/threads-max
```

在Linux上（不像大多数其他UNIX版本操作系统），线程和进程之间有一个映射，并且从OS的角度来看它们是相似的。

在Solaris上，这些参数可以在/etc/system文件中修改。这些修改不需要重新编译内核，但可能需要重新启动以便生效。我们可以通过修改max\_nprocs条目来修改进程的总数，而通过修改maxuproc来改变一个给定用户所允许的进程数目。

## 2. 文件描述符

不管何时，一个进程打开一个文件或一个socket的时候，就分配了一个叫做文件描述符的结构，直到文件关闭。OS限制了一个给定进程所能打开的文件描述符的数据，因而也限制了Web服务器所能同时拥有的连接的数目。如何修改这些设置，取决于操作系统。在Linux系统上，我们可以读取或修改/proc/sys/fs/file-max。在Solaris系统上，我们必须编辑/etc/system文件中的rlim\_fd\_max的值。修改后需要重新启动才能生效。

在<http://httpd.apache.org/docs/2.2/vhosts/fd-limits.html> 可以找到其他的信息。

## 3. 控制外部进程

Apache提供了几条指令来控制外部进程所使用的资源的数量。这样

的进程包括从服务器以及通过服务器端包含执行的程序产生的CGI脚本，但是，不包括那些作为模块调用的PHP脚本，因为模块是服务器进程的一部分。

**提示：**

按照本书的前几章的安装说明进行，将会使得PHP作为一个模块安装。因此，这些指令并不适用于你的情况，除非你修改自己的安装类型，或者处在一个虚拟主机环境下，其中的PHP并非作为一个模块安装的。然而，在后一种情况下，你很可能没有办法修改这些指令。

如下的Apache指令（用于httpd.conf中）只有在UNIX上可用，并且随着系统的不同而变化。

- **RLimitCPU**——接受两个参数：对一个进程所允许的CPU时间的秒数的软限制和硬限制。如果使用关键词**max**，表示操作系统所允许的最大设置。硬限制是可选的，软限制可以在重新启动之前更改，硬限制则为软限制设置指定了最大许可值。
- **RLimitMem**——语法和**RLimitCPU**相同，但是这一指令指定了每个进程所使用的内存的数量（以字节为单位）。
- **RLimitNProc**——语法和**RLimitCPU**相同，但是这一指令指定了进程的数目。

这3条指令对于预防恶意程序或编写很差的程序超出控制是很有帮助的。

### 29.1.2 和性能相关的Apache设置

本节给出影响性能的、不同的Apache设置。

## 1. 文件系统访问

从资源的角度来看，访问磁盘上的文件是一个昂贵的进程，因此应该尽量使服务一个请求所需的磁盘访问的数目最小化。符号链接、`per-directory`配置文件和内容协商是影响磁盘访问数目的一些因素。

- 符号链接——在UNIX中，符号链接（symbolic link，或symlink）是指向另一个文件的一类特殊的文件。它使用UNIX的`ln`命令创建，并且在使一个文件出现在不同的地方的时候很有用。

`Options`指令所允许的两个参数是`FollowSymLinks`和`SymLinksIfOwnerMatch`。默认情况下，Apache不会打开符号连接，因为它们可能绕过安全设置。因为，我们可以创建这样一个符号链接：从一个Web站点的公共部分链接到一个受限制的文件或目录，而文件和目录本来是不能通过Web打开的。因此，也是默认情况下，Apache需要执行检查来验证文件不是一个符号链接。如果`SymLinksIfOwnerMatch`存在，并且如果创建了符号链接的同一个用户拥有目标文件，它将打开一个符号链接。

由于这些测试必须对每个路径元素以及引用文件系统对象的每个路径执行，它们可能会增加系统负担。如果要控制内容创建，我们应该向配置添加一个`Options`为`+FollowSymLinks`的指令，并且避免`SymLinksIfOwnerMatch`参数。按照这种方法，测试不会发生，并且执行也不会受到影响。

- `per-directory`配置文件——正如第3章所介绍的，可能有`per-directory`配置文件。这些文件通常名为`.htaccess`，提供了配置服务器的一种

方便方式，并且允许某种程度的委托管理。

然而，如果要使这一功能可用，Apache必须在指向被请求文件的路径中的每个目录中查找这些文件，这会给文件系统访问带来沉重负担。如果你不需要per-directory配置文件，可以通过在配置中添加AllowOverride none来关闭这一功能。这么做将会避免因查找.htaccess文件而访问文件系统造成相关的性能损失。

- 内容协商——Apache可以根据客户机语言或偏好来提供一个文件的不同版本。这通过使用特定的和语言相关的扩展名来做到，但是，在这种情况下，Apache必须为每个请求而访问文件系统，查找具有这样的扩展名的文件。如果我们需要使用内容协商，确保至少使用一个类型映射文件，以最小化对磁盘的访问。为了国际化目的，基于Apache的内容协商的替代方法可以在第27章中找到。
- 记分板文件——这是一个特殊的文件，Apache主进程使用它来和旧操作系统上的子进程通信。我们可以使用ScoreBoardFile指令来指定其位置，但大多数现代操作系统都不需要使用这个文件。如果需要这个文件，可以发现将其放置在一个RAM磁盘上会提高性能。RAM磁盘是允许系统内存的一部分作为一个文件系统被访问的一种方法。创建一个RAM磁盘的细节根据系统而有所不同。

## 2. 网络 and 状态设置

一些和网络相关的Apache选项可能会降低性能。

- HostnameLookups——当HostnameLookups设置为on或double时，每次客户机发出一个请求，Apache会执行一个DNS查找来捕获客户机

的主机名。这种持续的查找将会导致响应过程的一个延迟，这条指令的默认设置是off。如果你想要捕获请求者的主机名，可以稍后用一个日志解析器来处理请求日志，离线地进行而不是实时地进行。

- 接受机制——Apache可以使用不同的机制来控制Apache子进程如何处理请求。优化的方法取决于特定的平台和进程数目。可以在<http://httpd.apache.org/docs-2.2/misc/perf-tuning.html> 找到相关信息。
- mod\_status——这个模块收集有关服务器、连接和请求的统计数据，这会降低Apache的速度。为了优化性能，关闭这一模块，或者至少确保ExtendedStatus设置为off，off也是其默认设置。



## 29.2 使用ApacheBench载入测试

我们可以使用测试工具和流量生成工具来测试站点的可扩展性和性能。有很多商业的和开源的工具可供使用，每个工具都具有不同程度的精度。一般来说，很难精确地模拟现实使用中的请求流量，因为访问者有不同的导航模式，使用具有不同速度的连接来访问Internet，如果花费时间太长就会停止下载，如果失去耐心就会重复地按下刷新按钮等等。同样的，一些工具记录实际网络流量供稍后的分析。

然而，为了更快而不是更精确地看看有关服务器处理大流量的能力的基本信息，Apache服务器提供了一个简单而有用的载入测试工具，叫做ApacheBench或ab。我们可以在Apache安装位置的/bin目录下找到它。

这个工具使我们能够多次请求某个URL并且显示出结果的一个概要。如下的命令使用10个客户机在给定的时间内同时请求www.example.com服务器的主页1000次。

```
# /usr/local/apache2/bin/ab -n 1000 -c 10 http://www.google.com/
```

### 提示：

如果我们调用ab而不使用任何参数，将会得到命令行选项和语法的一个完整列表。此外，目标URL末尾的斜杠是需要的，除非指定一个具体的页面。

运行结果如下所示。

This is ApacheBench, Version 2.3 <\$Revision: 655654 \$>  
Copyright 1996 Adam Twiss, Zeus Technology Ltd, <http://www.zeustech.net/>  
Licensed to The Apache Software Foundation, <http://www.apache.org/>

Benchmarking www.google.com (be patient)

Completed 100 requests  
Completed 200 requests  
Completed 300 requests  
Completed 400 requests  
Completed 500 requests  
Completed 600 requests  
Completed 700 requests  
Completed 800 requests  
Completed 900 requests  
Finished 1000 requests

Server Software: gws  
Server Hostname: www.google.com  
Server Port: 80

Document Path: /  
Document Length: 11955 bytes

Concurrency Level: 10  
Time taken for tests: 50.751 seconds  
Complete requests: 1000  
Failed requests: 669

(Connect: 0, Receive: 0, Length: 669, Exceptions: 0)

Write errors: 0  
Total transferred: 12710814 bytes  
HTML transferred: 11974814 bytes  
Requests per second: 19.70 [#/sec] (mean)  
Time per request: 507.515 - (mean)  
Time per request: 50.751 - (mean, across all concurrent requests)  
Transfer rate: 244.58 [Kbytes/sec] received

Connection Times (ms)

	min	mean[+/-sd]	median	max
Connect:	31	50 11.3	46	179
Processing:	88	454 53.8	449	803
Waiting:	84	285 119.0	282	694
Total:	136	504 56.5	499	850

Percentage of the requests served within a certain time (ms)

50%	499
66%	512
75%	521
80%	527
90%	540
95%	564
98%	711
99%	754
100%	850 (longest request)

这些请求都是通过Internet对一个示例服务器发出的。如果我们就在同一台计算机上的服务器或者本地局域网中的一个服务器执行测试，应该会得到每秒更多的请求。这个工具的输出一目了然，最相关的结果就是每秒请求的次数以及服务一个请求的平均时间。我们也可以看到所有的请求是如何在1秒以内得到服务的。

我们可以对请求数目以及同时请求的客户数使用不同的设置，来观察服务器在何时会有显著的性能下降。

## 29.3 预先性能调校

前面各节介绍了哪些设置可能会妨碍Apache的扩展，下面介绍一些预先增加服务器性能的技术。

### 29.3.1 把文件映射到内存

正如前面所介绍的，访问磁盘会显著地影响性能。尽管大多数现代操作系统为最为频繁访问的文件准备了一个缓存，Apache也允许我们显式地把一个文件映射到内存，从而不必再访问磁盘。执行这一映射的模块是`mod_file_cache`，我们可以通过`MMapFile`指令指定要映射到内存的文件的一个列表，它们一起作为一条完整的指令提交给服务器。Apache 2.x中的另一条指令`CacheFile`，接受一个文件列表，在启动的时候缓存文件描述符，并且在请求之间保存它们，从而节省了频繁地请求文件的时间和资源。

### 29.3.2 分布负载

另一种提高性能的方法是在数个服务器之间分布负载。这可以通过如下几种方法做到。

- 利用一个硬件负载均衡器在几个服务器之间导向网络和HTTP流量，使得它把单个服务器看作来自外部。
- 利用一个软件负载解决方案，它使用一个带有`mod_rewrite`的反向代理。
- 通过对服务器分工：提供图像的服务器、大的下载文件的服务器以

及其他静态内容的服务器。例如，可以把图像放在名为 `images.example.com` 的服务器上并且从主服务器链接它们。

### 29.3.3 缓存

服务内容的最快方法是根本不提供它。这可以通过如下方法来实现：在请求资源的时候，使用相应的HTTP标头来通知客户机和有效的代理。通过这种方法，一些出现在多个页面中的资源不再需要频繁地交换，例如logo或导航按钮，它们只在特定的一段时间内传送一次。

此外，我们可以使用Apache 2.x的`mod_cache`来缓存动态内容，这样它们就不必在每次请求的时候创建。这是一个潜在的性能显著提高点，因为动态内容通常需要访问数据库、处理模板等等，这会占用大量资源。

#### 提示：

Apache 2.4有很多缓存功能，当作是Apache之前版本的经验沿袭下来。要了解关于这一话题的更多信息，请参见位于<http://httpd.apache.org/docs/2.4/caching.html> 的Apache Caching Guide。

### 29.3.4 减少数据传输

减少服务器负载的另一种方法就是减少需要传送到客户机的数据量。这反过来也会使客户机的Web站点运行得更快，尤其是对那些使用较慢链接的站点。我们可以做一些事情来达到这一目的。

- 减少图像的数量。
- 减少图像的大小。

- 压缩大的、可下载的文件。
- 提前压缩静态HTML并且使用内容协商。
- 使用mod\_deflate压缩HTML内容。如果CPU的处理能力可用并且客户机通过较慢的链接，这将会很有用。内容将会快速发送，并且进程会很快应答其他的请求。

### 29.3.5 网络设置

HTTP 1.1允许通过一个连接服务多个请求，HTTP 1.0通过keep-alive扩展来实现这一点。KeepAliveTimeout指令使得我们能够指定服务器在关闭一个非活动的连接之前所等待的最大的秒数。增加超时时间意味着我们可以增加重用连接的机会。另一方面，它也在等待时间期间把连接和Apache进程联系起来，这可能妨碍本章前面所讨论的可扩展性。

## 29.4 防止滥用

拒绝服务（Denial of service, DoS）攻击同时使用大量的请求来淹没Web服务器，降低服务器速度或者阻止访问。一般来讲，DoS攻击很难预防，并且通常解决它们的最有效的方法在于网络级或操作系统级。一个例子就是阻止来自特定地址的对服务器的请求，尽管我们可以在Web服务器级阻止地址，但在网络防火墙、路由器或者使用操作系统网络过滤器阻止它们将会更有效。

滥用的其他类型包括发送超大请求或者同时打开多个链接。我们可以限制请求的大小和超时，从而使攻击效果最小化。默认的请求超时是300秒，但是，我们可以使用Timeout指令修改它。有多个指令可以用来控制请求主体和标头的大小，它们是LimitRequestBody、LimitRequestFields、LimitRequestFieldSize、LimitRequestLine和LimitXMLRequestBody。

机器人、Web蜘蛛和Web爬虫，都是用来描述访问Web站点中的页面的程序的名称，这些程序重复地打开站点链接的程序。Web搜索引擎使用这些程序来扫描Internet上的Web服务器，下载其内容并索引它们。现实生活中的用户使用这些程序来下载Web站点的全部或者部分，以便稍后离线的时候浏览。通常，这些程序行为良好，但是，有时候它们可能具有攻击性，并且用过多的同时连接来淹没你的网站，或者变成恶性循环。

行为良好的Web蜘蛛需要一个特别的文件，名为robots.txt，其中包

含了关于如何访问Web站点以及Web站点的哪部分不能供它们使用的说明。这个文件的语法可以在<http://www.robotstxt.org/> 找到。通过把一个格式正确的robots.txt文件放置到Web服务器的文档根目录下，我们可以控制蜘蛛的行为。此外，我们可以在路由器或操作系统级别停止请求。



## 29.5 实现虚拟主机

第一代Web服务器设计用来处理单个站点的内容。在同一机器上寄存数个Web站点的标准方式，是为每个站点安装和配置单独的Web服务器实例。随着Internet规模增大，寄存多个站点的需求也增长了，并且发展出了一种更为有效的解决方案：虚拟主机。虚拟主机允许Apache的一个单独实例来服务不同的站点，通过域名或者IP地址来识别它们。基于IP的虚拟主机意味着每个域名分配一个不同的IP地址；基于名字的虚拟主机意味着多个域名分享同一个IP地址。

Web客户机使用域名系统（domain name server system，DNS）来把主机名转换为IP地址，或者进行相反的转变。有如下几种映射方式。

- 一对一——每个主机名分配一个单独的、唯一的IP地址。这用于基于IP的虚拟主机中。
- 一对多——一个主机名分配给数个IP地址。这适合于多个Apache实例服务同一个Web站点。如果每个服务器安装在不同的机器上，就可能在它们之间平衡Web流量，提高可扩展性。
- 多对一——我们可以为数个主机名分配同一个IP地址。客户机将会在请求中使用Host: 标头来指定它要访问的Web站点。这用于基于名字的虚拟主机。

### 提示：

当使用多对一的映射的时候，一个DNS服务器通常可以配置为对每个DNS请求以一个不同的IP地址响应，这有助于分散负载，这就是轮循DNS。然而，如果你有机会使用一个负载

均衡设备而不是依赖一个DNS服务器，这么做将会缓解当我们把Web服务器绑定到DNS服务器时可能引发的任何问题。使用一个负载均衡器会消除Web服务器的高流量的可能性，也会降低DNS服务器的流量。

## 29.5.1 基于IP的虚拟主机

最简单的虚拟主机配置是为每个主机分配一个唯一的IP地址的时候。每个IP映射到HTTP请求，Apache负责把内容树分隔到它们自己的VirtualHost容器中，如下所示。

```
Listen 192.168.128.10:80
Listen 192.168.129.10:80

<VirtualHost 192.168.128.10:80>
    DocumentRoot /usr/local/apache2/htdocs/host1
    [other configurations specific to this host]
</VirtualHost>

<VirtualHost 192.168.129.10:80>
    DocumentRoot /usr/local/apache2/htdocs/host2
    [other configurations specific to this host]
</VirtualHost>
```

如果没有为一个给定的虚拟主机指定DocumentRoot，那么在<VirtualHost>段外指定的全局设定将被使用。在前面的例子中，每个虚拟主机有自己的DocumentRoot。当一个请求到达，Apache使用目标IP地址来把请求导向相应的主机。例如，如果一个请求来自IP 192.168.128.10，Apache返回来自/usr/local/apache2/htdocs/host1的文档。

如果主机操作系统没有使用VirtualHost容器的名字解析一个IP地址，并且没有ServerName指令，Apache将在服务器启动的时候报告它无法把IP地址映射为主机名。这个报告并不是一个重要的错误。Apache仍将运行，但是这个错误指示应该对DNS配置做一些工作，以便Web浏览

器可以找到服务器。可以使用一个全称域名（fully qualified domain name, FQDN）而不是一个IP地址来作为VirtualHost容器的名字以及Listen指令的绑定（如果在DNS中一个域名解析为一个在机器上配置的IP地址，并且Apache可以绑定到它）。

## 29.5.2 基于名字的虚拟主机

作为使用虚拟主机时减轻对IP地址消耗的一种方法，HTTP/1.1协议版本引入了Host: 标头，它使得浏览器能够指定请求所要求的具体主机。这就允许多个主机分享同一个单个的IP地址。大多数浏览器现在提供对HTTP/1.1的支持。

不能对基于名称的虚拟主机使用SSL，除非在严格控制的情况下。请参见[http://httpd.apache.org/docs/2.4/ssl/ssl\\_faq.html#vhosts](http://httpd.apache.org/docs/2.4/ssl/ssl_faq.html#vhosts) 以了解更多信息。然而，你可以对基于IP的虚拟主机使用SSL。

### 提示：

尽管Host: 的使用已经标准化到了HTTP/1.1规范中，但一些较旧的HTTP/1.0浏览器也支持这一标头。

程序清单29.1展示了来自Mozilla Firefox浏览器的一组典型的请求标头。此外，如果URL和一个端口号一起输入，它也可以是Host标头内容的一部分。

程序清单29.1 请求标头

---

```
GET / HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.7 (KHTML, like Gecko)
  Chrome/16.0.912.75 Safari/535.7
Host: host1.example.com
Connection: Keep-Alive
```

---

**Apache使用Host:** 标头来配置哪几个主机名可以通过一个IP地址共享, 即本章前面所概括的多对一的情况, 因此, 叫做基于名字的虚拟主机。

对于Apache 2.4之前的版本, 使用NameVirtualHost指令可以指定IP地址和端口的组合, 服务器正是在这一组合上接受基于名称的虚拟主机请求的。在Apache 2.2及更早的版本中, 这是用于基于名称的虚拟主机的一条必需指令。

**提示:**

如果你使用Apache 2.4, 就不能使用NameVirtualHost指令。相反, 只要确保在VirtualHost容器中匹配了正确的IP以及ServerName。

程序清单29.2使得Apache根据Host标头的内容把所有的连接分配到192.168.128.10。

程序清单29.2 基于名字的虚拟主机

---

```
#Only use NameVirtualHost if pre-Apache 2.4
NameVirtualHost 192.168.128.10
Listen 192.168.128.10:80

<VirtualHost 192.168.128.10>
    ServerName host1.example.com
    DocumentRoot /usr/local/apache2/htdocs/host1
</VirtualHost>

<VirtualHost 192.168.128.10>
    ServerName host2.example.com
    DocumentRoot /usr/local/apache2/htdocs/host2
</VirtualHost>
```

---

对于解析为192.168.128.10的每个主机名，Apache可以支持另一个基于名字的虚拟主机。如果来自于该IP地址的请求所针对的主机名没有包含在配置文件之中，例如host3.example.com，Apache只是把请求关联到配置文件中的第一个容器，在这个例子中就是host1.example.com。对于没有带一个Host标头的请求，也适用这一行为，配置文件的第一个容器，就是获取请求的那一个容器。

来自example.com域的一个最终用户可能让自己的机器把example.com设置为默认域名。在这种情况下，他可能把浏览器导向<http://host1/>，而不是全称域名<http://host1.example.com/>。Host标头内将只有host1，而不是host1.example.com。为了确保正确的虚拟主机容器得到请求，我们可以使用程序清单29.3中所示的ServerAlias指令。

程序清单29.3 ServerAlias指令

---

```
#Only use NameVirtualHost if pre-Apache 2.4
NameVirtualHost 192.168.128.10
Listen 192.168.128.10:80

<VirtualHost 192.168.128.10>
    ServerName host1.example.com
    ServerAlias host1
    DocumentRoot /usr/local/apache2/htdocs/host1
</VirtualHost>

<VirtualHost 192.168.128.10>
    ServerName host2.example.com
    ServerAlias host2
    DocumentRoot /usr/local/apache2/htdocs/host2
</VirtualHost>
```

---

实际上，我们可以给ServerAlias一个空格分开的名字的列表，这些名字可能出现在Host标头中，这样，我们就不需要一个单独的VirtualHost容器，以及很多只是用来处理所有名字变量的常见指令。

HTTP 1.1强制使用Host标头。如果HTTP请求行中的协议版本是1.1，请求必须带有一个Host标头。在基于名字的虚拟主机的早期，把Host标头当作一种折中措施：需要较少的IP资源，但是不能发送Host标头的遗留浏览器仍然可以使用，因此，无法访问所有的服务器虚拟主机。今天，不再考虑这一点，因为还在使用的这样的遗留浏览器已经为数不多了。

### 29.5.3 大量虚拟主机

在前面的列表中，DocumentRoot指令遵从如下一个简单的模式。

```
DocumentRoot /usr/local/apache2/htdocs/hostname
```

其中hostname就是用于虚拟主机的ServerName中的全称域名的主机名部分。对于只有少数几个虚拟主机的情况，这个配置很好。但是，如

果有数十个、数百个甚至数千个虚拟主机，该怎么办呢？这个配置文件会变得很难维护。Apache为带有mod\_vhost\_alias的同样的虚拟主机提供了一种有效的解决方案。我们可以在VirtualDocumentRoot指令中配置Apache，把虚拟主机请求映射到分开的带有模式匹配规则的内容树中。这一功能对于想要为每个用户提供一个虚拟主机的ISP特别有用。如下的例子提供了一个简单的大量虚拟主机配置。

```
#Only use NameVirtualHost if pre-Apache 2.4
NameVirtualHost 192.168.128.10
Listen 192.168.128.10:80

VirtualDocumentRoot /usr/local/apache2/htdocs/%1
```

这个例子的VirtualDocumentRoot指令中的%1标记将由全称域名的第一部分替代。mod\_vhost\_alias指令拥有一种语言用来把FQDN组成部分映射为文件系统位置，包括FQDN中的字符。

如果我们删除所有的VirtualHost容器，并且把配置简化为上述配置，服务器将为创建于/usr/local/apache2/htdocs目录下的任何子目录的访问请求服务。如果FQDN的主机名部分和一个子目录匹配，Apache在将请求翻译为一个文件系统位置的时候，将在那里查找内容。

尽管虚拟主机通常从主服务器环境继承命令，但其中的一些，例如Alias指令，不会被继承。例如，虚拟主机将不会继承如下的文件系统映射。

```
Alias /icons /usr/local/apache2/icons
```

Options指令的FollowSymLinks标志在这个环境中也不可用。然而，ScriptAlias指令的另一种形式得到支持。

如下展示的VirtualScriptAlias指令把对/cgi-bin下的任何资源的请求当作包含的CGI脚本。

```
#Only use NameVirtualHost if pre-Apache 2.4
NameVirtualHost 192.168.128.10
Listen 192.168.128.10:80
VirtualDocumentRoot /usr/local/apache2/htdocs/%1/docs
VirtualScriptAlias /usr/local/apache2/htdocs/%1/cgi-bin
```

注意，cgi-bin是该指令的一个特殊标记，将该目录只叫做cgi是不能工作的，必须是cgi-bin。

为了满足基于IP的虚拟主机的需求，这些指令都有其他的形式：Virtual-DocumentRootIP和VirtualScriptAliasIP。



## 29.6 小结

本章提供了有关Apache和操作系统设置的信息，它们可能影响到可扩展性和性能。在大多数情况下，Web站点可扩展性的问题关系到动态内容的生成和数据库访问。编写有效的脚本将会缓解这些问题。和硬件相关的改善，例如高质量网卡和驱动器，增加的内存以及磁盘阵列也可以提供增强的性能。

关于虚拟主机，Apache可以以多种方式配置来处理虚拟主机。不管你需要大量的同样的虚拟主机，还是一组不同的虚拟主机配置，或者是可用的IP地址的数目受到限制，都有一种方法可以为你的应用程序配置Apache。基于名字的虚拟主机是部署虚拟主机而不需要用尽IP地址的一种常用技术。当我们有足够的IP地址并且我们想要配置尽可能地简单的时候，基于IP的虚拟主机是另一种常见方法。另外，如果我们不能修改DNS的配置，可以为虚拟主机而使用单独的端口号的资源。

## 29.7 Q&A

**Q:** 如何测量我的站点是否足够快？

**A:** 很多开发者在本地或者通过一个内部网络来测试他们的站点，但是，如果在一个公共的Web站点运行，测试会更好，因为许多的用户将能够通过慢速连接访问它。尝试从一个拨号账号去访问你的网站，并且确保页面载入足够快。首要的原则是，页面应该在3秒内载入。

**Q:** 如何把一个已有的基于名字的虚拟主机迁移到它自己的机器上，而同时维持其服务？

**A:** 如果一个虚拟主机的目标是移动到邻近的机器，该机器在定义上不能具有相同的IP地址，有一些额外的方法可以采取。常见的做法像下面这样，尽管在具体步骤上可能有多种形式。

1. 把DNS映射的存留时间（time-to-live, TTL）设置为一个非常低的值。这会增加客户机的主机名查找的频率。
2. 在具有新的IP地址的旧主机上，配置一个IP别名。
3. 配置虚拟主机的内容既由基于名字的虚拟主机提供服务，也由基于IP地址虚拟主机提供服务。
4. 在对于旧IP地址的虚拟主机的所有请求消失后（由于对DNS缓存的旧查找过期了），可以迁移服务器。

**Q:** 我可以混用基于**IP**的虚拟主机和基于名字的虚拟主机吗？

**A:** 是的。如果绑定了多个IP地址，你可以用多种方式来分配其用法。一组基于名称的虚拟主机可能和每一个IP关联，只要对每一个IP使用一条单独的NameVirtualHost指令（如果是Apache 2.4之前的版本的话），或者是使用一组控制的ServerName指令。例如，一个IP可能专门用作SSL的、基于IP的虚拟主机，而另一个IP可能用于一组基于名称的虚拟主机。

## 29.8 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 指出可能限制可扩展性或影响Apache性能的Apache设置。
2. 指出可能限制可扩展性的某些操作系统设置。
3. 指出提高性能的某些方法。
4. 一个VirtualHost容器中的ServerNamedirective是必需的吗？

## 解答

1. 一些Apache设置可能影响到可扩展性，包括Options指令的FollowSymLinks和SymLinksIfOwnerMatch参数，启用per-directory配置文件，主机名查找，拥有一个记分板文件以及带有mod\_status的统计集合。
2. 一些操作系统设置可能影响到可扩展性，包括进程数目的限制、打开文件描述符以及每个进程允许的内存。
3. 下面是提高性能的一些建议：通过一个硬件负载均衡器或反向代理来进行负载分布、数据压缩、缓存、把文件映射到内存，以及静态编译模块。
4. 只有当使用基于名称的虚拟主机的时候，一个VirtualHost容器中的ServerNamedirective才是必需的。主机标头内容会和ServerNamedirective的内容进行比较。如果匹配不成功，会检查VirtualHost容器的ServerAlias指令值是否匹配。

## 第30章 建立一个安全的Web服务器

在本章，我们将学习：

- **SSL/TLS**协议族和底层的加密概念。
- 什么是安全认证，以及如何创建和管理它们。
- 如何在**Apache 2.2**中激活**mod\_ssl** **Apache**模块。

本章介绍如何建立一个能够安全工作的**Apache**服务器。如果你使用主机提供商的一个共享服务空间，你就不必了解这些设置选项，因为设置一个安全的**Web**服务器的过程已经由提供商来完成，通常不需你亲自动手去做。然而，如果你具有自己的服务器的**root**访问权限，本章将向你介绍如何使自己的**Web**服务器更加安全。

## 30.1 安全性的需求

几种和Internet相关的事务都需要较高级别的安全性。这包括金融事务，例如银行运作和电子商务，还有敏感信息的任何交换，例如医疗记录和公司文档。Internet上的安全事务需要如下3个主要元素。

- 机密性——如果你在传输或访问诸如信用卡号或者个人医疗病历这样的敏感信息，肯定不希望陌生人能够获取它们。
- 完整性——传送的信息必须保证不会被外界修改。如果你在线下一个订单，购买100股的股票，不希望任何人能够获取这些消息，并且将订单修改为购买1000股。
- 验证——需要确信我们所通讯的组织或者个人就是他们自己所表明的身分。



## 30.2 SSL协议

SSL表示Secure Sockets Layer（安全套接字层），而TLS表示Transport Layer Security（传输层安全）。这两个协议族最初设计用来为HTTP事务提供安全性，但是它们也用于各种其他的Internet协议，如IMAP（Internet Message Access Protocol，互联网邮件访问协议）和NNTP（Network News Transfer Protocol，网络新闻传输协议）。在SSL上运行的HTTP叫做安全HTTP。

Netscape在1994年发布了SSL版本2，并且在1995年发布了SSL版本3。TLS是一个IETF标准，它设计用来把SSL标准化为一个Internet协议，但是，它只是SSL版本3的一个修改，进行了少量的功能增加和清理。TLS这个缩略语是Microsoft和Netscape在协议的命名上进行争论后的结果，因为每家公司都想使用自己的名字。然而，这个名字并没有流传下来，大多数人还是简单地把这个协议叫做SSL。除非单独声明，本章剩余的部分都把SSL/TLS称作SSL。

通过在URI（Uniform Resource Identifier，统一资源标识符）中用https来替换掉http，我们表示希望使用SSL连接到一台服务器。SSL的HTTP的默认端口是443。

下面一节介绍了SSL如何解决前面所列出的保密性、完整性和验证需求。我们还了解了SSL核心的一些底层的数学和加密原则。

### 30.2.1 解决保密性需求

SSL协议通过加密数据来保护它们。加密是一个转换消息的过程，把明文转换为一个新的加密的消息，也就是密文。尽管每个人都可以读懂明文，但是，密文对于可能截取它的每个人来说都是无法了解的。解密是相反的过程，会把密文转换回最初的明文。

通常，加密和解密的过程都涉及到一段额外的信息，即密钥。如果发送者和接受者都共享同样的密钥，这个过程叫做对称加密（symmetric cryptography）。如果发送者和接受者拥有不同的密钥，即补充密钥，这个过程叫做非对称加密（asymmetric cryptography）或公钥加密（public key cryptography）。

## 1. 对称加密

如果用于加密和解密的密钥是相同的，这个过程叫做对称加密。DES、Triple-DES、RC4和RC2都是用于对称密钥加密的算法。大多数这些算法可以拥有不同的密钥大小，以位为单位。通常，给定一个算法，密钥的位数越大，加密越安全，但它运行得也越慢，因为执行算法所需的计算量增加了。

对称加密和公钥加密相比要快一些，我们将在下一节介绍后者。然而，对称加密有两个主要的缺陷。一个是密钥必须定期更改，以防止可能有窃听者通过同样的密钥访问大量加密的材料。另一个问题是密钥的分发问题：如何把密钥发送给团体中的每一个人，而且是以安全的方式呢？这就是对称加密的一个最初的局限因素，这个问题通过定期让人带着装满密钥的手提箱周游一遍而解决。接下来看看公钥加密。

## 2. 公钥加密

公钥加密采取和对称加密不同的方法。存在一对密钥，一个是公用的，一个是私有的，而不是共享相同的密钥。公钥可以广泛地发布，而所有者秘密地保持私钥。这两个密钥互为补充，用一个密钥加密的消息只能被另一个密钥解密。

使用这种方法，想传送一条安全消息给你的任何人，可以使用你的公钥来加密这条消息，这就确保了只有私钥的拥有者，也就是你，才能解密它。即便一个窃听者获得了公钥，他也无法解密你们交流的消息。实际上，你需要公钥尽可能广泛地使用，这样，更多的人可以发送加密的消息给你。公钥加密也可以用来提供消息的完整性并验证消息。具有公钥的人将会把这些密钥放在公钥服务器上，或者直接把公钥发送给那些想要使用安全邮件交换的人们。使用相应的软件工具，例如PGP或GnuPG，发送者可以根据接受者的公钥来加密外发的消息。

只有私钥的拥有者才能解密发送给自己的消息，这一断言是根据当前密码学的知识以及可用的计算能力而得到的，直接暴力破解将不会在一个合理的期限内破解密码。然而，如果底层的算法或者其实现有缺陷，这样的攻击也是有可能的。

**提示：**

公钥加密类似于发送很多可交换锁，但保留了主密钥。想要给你发送一条私密消息的任何人可以这样做：在发送之前保证其安全并用这些锁（公钥）中的一个来锁定它。只有有相应的钥匙（私钥）的人才能打开这个锁（解密消息）。

SSL协议在最初的握手阶段使用公钥加密来确保安全地交换对称密钥，此后，对称密钥可以用来加密通讯。

### 30.2.2 解决完整性的需求

数据完整性通过如下方式来保证：在消息的内容上执行一个特殊的计算并且把结果和消息本身保存起来。当消息到达目的地时，接受者可以执行相同的计算并且比较结果。如果消息的内容发生了变化，计算的结果会不同，这样我们就知道有别的人改动了消息。

摘要算法也执行这一过程来创建消息摘要。针对任意一条消息创建一个固定长度的表示，这可以像指纹一样唯一地标识该消息，这种方法就叫做消息摘要（**message digest**）。一个好的消息摘要算法应该是不可逆的而且可以避免冲突，至少从实用的目的是这样。

不可逆意味着无法从摘要得出最初的消息，而避免冲突意味着不会有两条不同的消息拥有相同的摘要。常见的摘要算法是MD5和SHA。

然而，只有消息摘要，无法保证消息的完整性，攻击者可以改变文本和消息摘要。消息验证码（**message authentication code, MAC**）类似于消息摘要，但是在该过程中加入了一个共享的秘密密钥。算法的结果取决于所使用的消息和密钥。由于攻击者无法访问密钥，他无法同时修改消息和密钥。HMAC（**Hash Message Authentication Code, Hash消息验证码**）是消息验证码算法的一个例子。

SSL协议使用MAC代码来避免重播攻击，并且确保了发送信息的完整性。

### 30.2.3 解决验证的需求

SSL使用认证来验证通信中的团体。公钥加密可以用来数字化地标

记消息。实际上，仅仅使用密钥加密消息的做法，向接收者保证了消息是来自于你。其他的数字签名算法包括各先计算消息的一个摘要，然后再标记该摘要。

我们可以分辨出创建了公钥和私钥对的人就是发送消息的人，但是，如何把密钥和你在现实世界中所信任的一个人或组织联系起来呢？一个攻击者可能模仿一个发送者的身份并且发布一个不同的公钥，宣称这就是合法的公钥，这也是有可能的。

信任可以通过使用数字证书来实现。数字证书就是包含了一个公钥和关于其所有者的信息（名字、地址等等）的电子文档。为了有用处，证书必须由一个可信任的第三方机构签署（即认证机构，**certification authority, CA**），认证机构来认证信息是正确的。正如本章稍后所介绍的，有很多种不同的CA。其中的一些是商业实体，为通过Internet开展业务的公司提供认证服务。提供国内认证服务的公司创建其他的CA。

CA保证证书中的信息是正确的，并且该密钥属于个人或组织。证书也有一个有效时期，并且可能过期或取消。证书可以传递，因此，认证过程可以委托。例如，一个可信的实体可以认证公司，反过来，公司可以负责认证自己的雇员。

如果整个这个过程是有效的并且是可信的，在认证机构颁发一个证书之前，它必须要求有来自个人或组织的相应的身份证明。

默认情况下，浏览器包含了一组用于可信的认证机构的根证书。

## 1. SSL和证书

定义证书的主要标准是X.509，适合于Internet使用。一个X.509证书包含以下信息。

- 颁发者——证书签名者的名字。
- 主体——保留认证的密钥的人。
- 主体公钥——主体的公钥。
- 控制信息——像证书的有效日期之类的数据。
- 签名——包含前面的数据的签名。

我们可以通过浏览器连接到一个安全的服务器来检查一个真实的证书。如果连接成功，一个小小的锁图标或者另一可见的线索将会添加到浏览器的状态栏上。根据浏览器的不同，我们应该能够单击表示的图标来查看有关SSL连接和远程服务器证书的信息。在下面的例子中，SSL证书在<http://www.amazon.com/>可以查看到。我们可以看到证书的颁发者是GlobalSign（如图30-1所示）。这个页面无缝地下载，因为GlobalSign是一个可信的认证机构。



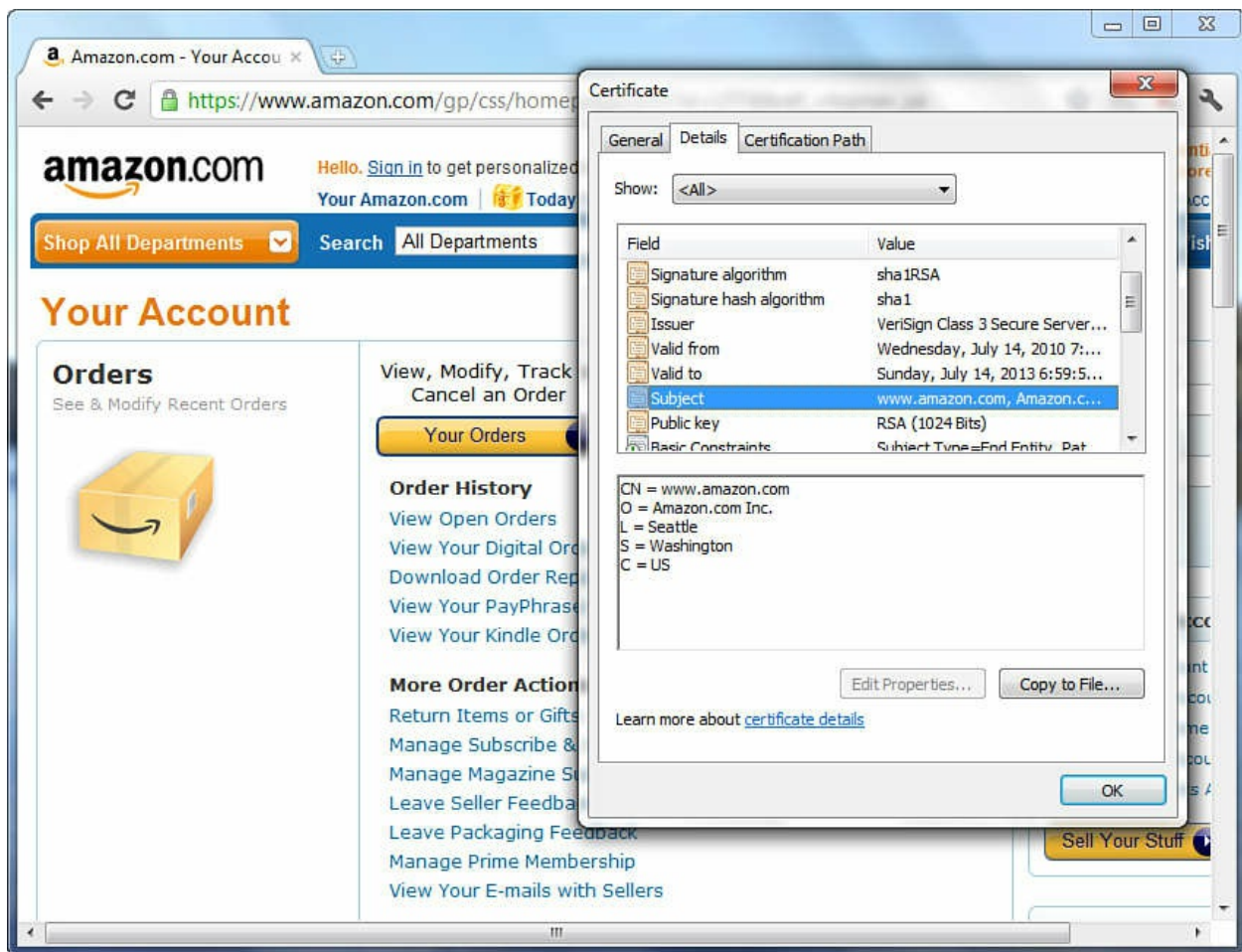


图30-1 www.amazon.com所使用的SSL证书

在图30-1所示的认证中，Subject是名为Amazon.com，Inc的一个实体，位于美国华盛顿西雅图。这个实体常用的名称是www.amazon.com。

C表示国家，ST表示州，L表示地点，O表示组织，CN是常用名字。在一个Web站点的证书例子中，常用名字表示该Web站点的全称域名，这就是URL的服务器名部分。在这个例子中，就是www.amazon.com域和主机名“www”。如果这个和我们在地址栏输入的不一致，浏览器将会发出一个错误。

如果CN是\*.amazon.com，那么对于amazon.com域上的任何主机，SSL验证都是有效的。

## 2. SSL协议总结

我们已经看到了SSL如何通过加密实现保密性，通过消息验证码实现完整性，以及通过证书和数字签名实现验证。建立一个SSL连接的过程如下。

1. 用户使用自己的浏览器连接到远程Web服务器。
2. 握手阶段开始，浏览器和服务器交换密钥和证书信息。
3. 浏览器检查服务器证书的有效性，包括它是否过期，是否由一个可信的CA颁发，等等。
4. 可选地，服务器可以要求客户机也提供一个有效的证书。
5. 服务器和客户机使用彼此的公钥来安全地达成一个对称密钥。
6. 握手阶段结束，并且继续使用对称加密传输数据。



## 30.3 获取和安装SSL工具

对SSL的支持是由一个名为mod\_ssl的Apache模块所提供的。这个模块需要OpenSSL库，这是SSL/TLS协议的一个开源实现，而且还具有各种其他的加密算法。OpenSSL基于Eric A.Young和Tim J. Hudson开发的SSLey库。

由于世界范围内对字符串加密发布的限制以及知识产权的保护，SSL相关工具的安装随着平台不同而难易程度不同。下面的小节提供了获取和安装SSL相关工具的一个概览。

### 30.3.1 OpenSSL

安装OpenSSL所需的所有文件和说明可以在<http://www.openssl.org/>找到。UNIX/Linux（及其变体）的用户将会发现，OpenSSL软件的安装和其他系统工具的安装类似。然而，Windows用户将会发现，目前还没有免费发布的预编译二进制代码可用。因此，Windows用户必须自行编译OpenSSL工具。安装了OpenSSL工具箱之后，将会拥有创建和操作证书和密钥以及和mod\_ssl Apache模块交互的所有必需的元素。

#### 1. Windows用户的安装

熟悉构建自己的二进制的过程的Windows用户可以通过OpenSSL站点所提供的OpenSSL源代码来自行完成。在Windows上编译OpenSSL的说明可以在源发布中的INSTALL.W32文件中找到。重复这些说明已经超出了本书的范围，然而，你可以发现，它们编写得全面而周到。所需的工具是ActiveState Perl for Windows，以及如下的任一款C编译器。

- Visual C++
- Borland C
- GNU C (Cygwin or MinGW)

确保按照说明来选择相应的编译器，因为它们彼此有所区别。可以在[http://httpd.apache.org/docs/2.2/platform/win\\_compiling.html](http://httpd.apache.org/docs/2.2/platform/win_compiling.html) 找到从Apache编译OpenSSL的技巧。

## 2. UNIX/Linux用户的安装

如果你运行的是Linux或FreeBSD的最近发布版本，OpenSSL可能已经在你的系统上安装了。如果你需要安装OpenSSL，可以从OpenSSL站点下载源文件。下载文件之后，解压缩它并且cd到创建的目录（把下列指令中的-version替换成你自己特定的OpenSSL当前版本），命令如下。

```
# gunzip < openssl-version.tar.gz | tar xvf -  
# cd openssl-version
```

完整的安装说明可以在INSTALL文件中找到。简而言之，config脚本将帮助你编译软件，后面跟着就是make和make install过程。

### 30.3.2 Apache的mod\_ssl模块

过去，Apache的SSL扩展必须单独发布，因为导出受限制。现在，mod\_ssl绑定到了Apache 2.2，但只是作为源发布的一部分。尽管这对于UNIX/Linux用户不是问题，但是，Windows用户将会发现必须从源代码编译Apache以生成mod\_ssl模块，mod\_ssl没有在预编译和发布的二进制版本中发布。mod\_ssl模块依赖于OpenSSL库，因此，有效的OpenSSL安装也是必需的。

## 1. 对于Windows用户

当下载预编译的安装二进制文件的时候，确保选择文件名中带有“openssl”的版本。例如，httpd-2.2.22-win32-x86-openssl-0.9.8t.ms就是针对Windows的Apache 2.2.22安装程序。

在编写本书时，针对Apache 2.4还没有Windows二进制发布版。有这个版本的时候，它应该会遵从这一命名惯例。

如果你想使用mod\_ssl从源构建OpenSSL和Apache，请查看位于[http://httpd.apache.org/docs/2.4/platform/win\\_compiling.html](http://httpd.apache.org/docs/2.4/platform/win_compiling.html)的Apache文档。重复这些说明超出了本书的范围，但它们将会为你提供所需的全部信息。核心的要求如下所示。

- 安装OpenSSL工具箱。
- Microsoft Visual C++ 5.0或更高版本。
- Windows平台SDK。
- awk工具（awk、gawk或类似工具）。

## 2. 对于UNIX/Linux用户

第3章中使用的源发布应该已经包含了使用mod\_ssl所需的文件。因此，对于使用mod\_ssl的UNIX/Linux用户，只需要再次执行配置过程和make/make install过程，并添加如下内容作为配置命令的一部分。

```
--enable-ssl --with-ssl=/usr/local/ssl/
```

这里假设你已经在列出的位置安装了OpenSSL，如果它位于服务器上其他的目录，只要在上述的命令中用正确的位置替换就行了。如果静

态地把mod\_ssl编译到Apache，可以通过执行如下的命令检查它是否存在，该命令提供一个编译进的模块的列表。

```
# /usr/local/apache2/bin/httpd -l
```

提示：

上述命令假设我们把Apache安装到了/usr/local/apache2目录。

如果mod\_ssl作为一个动态加载模块编译，如下的代码行必须添加到Apache的配置文件中(httpd.conf)。

```
LoadModule ssl_module modules/libmodssl.so
```

完成了对httpd.conf文件的改变之后，重新启动Apache以使修改生效。如果在重新启动之后查看error\_log，mod\_ssl将是服务器签名的一部分，如下所示。

```
Apache/2.4.1 (Unix) mod_ssl/2.4.1 OpenSSL/0.9.8t PHP/5.4.0
```

## 30.4 管理证书

在安装和配置了OpenSSL和mod\_ssl之后，一个可工作的SSL服务器需要实现的下一步就是创建一个服务器证书。本节详细说明如何使用openssl命令行工具来创建和管理证书和密钥。如果为一个电子商务站点使用SSL，加密将会保证客户数据不被窃取，并且证书确保了客户可以验证其身份正如自己所表明的那样。

### 提示：

下面的例子引用了命令程序openssl的UNIX版本。如果我们在Windows下运行，需要使用openssl.exe代替，并且把例子中的路径改用反斜杠而不是斜杠。另外，如果我们安装OpenSSL的路径和这里给出的不一样，直接替换例子中的目录就可以了。

### 30.4.1 创建一个密钥对

在创建一个证书请求之前，必须拥有一个公钥/私钥对。假设我们要创建的证书的全称域名（FQDN）是www.example.com。我们可以通过执行如下命令来创建密钥。

```
# openssl genrsa --des3 -out www.example.com.key 1024
```

- genrsa开关向OpenSSL表示我们要生成一个键值对。
- des3开关表示，私钥应该加密并且用一个密码来保护。
- out开关表示结果存储到哪里。
- 1024表示所生成的密钥的位数。

调用这条命令的结果如下所示。

```
Generating RSA private key, 1024 bit long modulus
.....++++++
.....++++++
e is 65537 (0x10001)
Enter pass phrase for www.example.com.key:
```

正如你所见到的，要求我们提供一个密码，选择一个安全的密码。密码不是保护私钥所必需的，但是，无论何时当你想要启动服务器的时候，都会被询问密码。

提示：

我们可以选择没有密码保护的密钥。这很方便，因为我们不需要在重新启动的时候输入密码，但是这很不安全，并且服务器的不安全也意味着密钥的不安全。在任何情况下，我们可以通过在生成命令中省略-des3开关或者通过执行如下的命令来不保护密码。

```
# openssl rsa -in www.example.com.key -out www.example.com.key.unsecured
```

备份www.example.com.key文件是一个好主意。你可以通过执行如下命令来了解密钥文件的内容。

```
# openssl rsa -noout -text -in www.example.com.key
```

密钥的元素随后将会显示给你。

### 30.4.2 创建一个证书签发请求

要获得CA颁发的一个证书，你必须提交一个证书签发请求。要创建一个请求，执行如下命令。

```
# req -new -key www.example.com.key -out www.example.com.csr
```

上述命令将会提示你输入证书信息，如下所示。

```
Using configuration from /usr/local/ssl/install/openssl/openssl.cnf
Enter PEM pass phrase:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:CA
Locality Name (eg, city) []: San Francisco
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (eg, YOUR name) []:www.example.com
Email Address []:administrator@example.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Common Name字段的输入要和你的Web站点的访问者在他们的浏览器中输入的内容一致，这非常重要。这是浏览器对于远程服务器证书所要执行的检查之一。如果名字不同，将会有有一个警告提示用户地址不匹配。

现在证书已存储在www.example.com.csr。我们可以使用如下的命令来了解证书的内容。

```
# openssl req -noout -text -in www.example.com.csr
```

我们可以向一个CA提供证书签发请求以供处理。VeriSign、Thawte和Network Solutions就是这样的CA，当然还有很多CA可供使用。我们可以通过站点了解关于VeriSign、Thawte和Network Solutions的提交程序及其产品的更多信息。

- VeriSign—<http://www.verisign.com/ssl/>

- Thawte—<http://www.thawte.com/ssl/web-server-ssl-certificates/>
- Network Solutions —<http://www.networksolutions.com/SSL-certificates/>

### 30.4.3 创建一个自签发的证书

我们也可以创建一个自签发的证书。也就是说，可以既是颁发者也是证书的主体。尽管这对于一个商业站点不是很有用，但它还是使我们能够在等待来自CA的正式证书的同时，测试自己的mod\_ssl安装并拥有一个安全的Web服务器。

```
# openssl x509 -req -days 30 -in  
www.example.com.csr -signkey  
www.example.com.key -out www.example.com.cert
```

我们需要把证书www.example.com.cert（要么是CA所返回的，要么是自己签发的）复制到/usr/local/ssl/openssl/certs/，并且把密钥复制到/usr/local/ssl/openssl/private/。

通过执行如下的命令来保护密钥文件。

```
# chmod 400 www.example.com.key
```

这条命令使得密钥只能供root用户阅读。



## 30.5 SSL配置

前面的各节介绍了SSL背后的（不是非常基础的）概念，并且我们学习了如何生成密钥和证书。现在，我们可以配置Apache以支持SSL了。正如在本章前面学过的，`mod_ssl`模块要么必须静态地编译，要么将其编译为一个可载入的模块，这样`httpd.conf`文件中必须有相应的`LoadModule`指令。

Apache 2.2.x带有一个“额外的”配置文件，专门用来运行一台支持SSL的服务器。要使用这个额外文件，只需要去掉`httpd.conf`中的如下这行注释。

```
Include conf/extra/httpd-ssl.conf
```

接下来，修改`httpd-ssl.conf`中的标准配置代码段。当然，需要用你自己的信息加以替换。

```
UseCanonicalName On
<VirtualHost www.example.com:443>
ServerName www.example.com
SSLEngine on
SSLCertificateFile /usr/local/ssl/openssl/certs/www.example.com.cert
SSLCertificateKeyFile /usr/local/ssl/openssl/certs/www.example.com.key
</VirtualHost>
```

这个代码段设置了一个新的虚拟主机，它将监听443号端口（HTTPS的默认端口），我们使用`SSLEngine`命令在这个虚拟主机上打开SSL。`SSLCertificateFile`和`SSLCertificateKeyfile`指令表示服务器证书在哪里以及包含相关密钥的文件在哪里。

### 启动服务器

当我们想要以安全模式启动Apache的时候，之前的Apache版本都要求执行一条`apachectl startssl`命令。然而，按照上一节中所定义的配置方法，以安全模式启动Apache和不带SSL启动Apache没什么区别：执行`apachectl start`命令就行了。只要通过指令把`httpd-ssl.conf`文件包含在`httpd.conf`中，每次使用前面的指令，Apache都会启动支持SSL的服务器。

如果服务器正在运行并且我们重新启动它，同时如果密钥是受保护的，将会提示我们输入密码。在输入了正确的密码之后，Apache将会启动，并且我们可以使用类似`https://www.example.com/`的URL安全地连接到它。当然，用你自己的域名替代。如果我们无法成功地启动服务器，检查Apache错误日志以找到有关出错的线索。例如，如果我们无法绑定到端口，确保其他的Apache实例没有运行。我们必须拥有管理员权限才能绑定到443端口。

## 30.6 小结

本章介绍了SSL协议和mod\_ssl的基础，mod\_ssl是实现对SSL支持的Apache模块。首先介绍了OpenSSL和mod\_ssl的安装和配置，以及如何使用openssl命令行工具来实现证书和密钥的生成和管理。你可以访问位于[http://httpd.apache.org/docs/2.4/mod/mod\\_ssl.html](http://httpd.apache.org/docs/2.4/mod/mod_ssl.html)的mod\_ssl参考文档来了解深入的语法说明和其他的配置信息。别忘了，SSL只是维护一个安全的服务器的一部分，后者还包括应用安全补丁、OS配置、访问控制、物理安全性等等。

## 30.7 Q&A

**Q:** 我可以让基于名字的虚拟主机使用**SSL**吗？

**A:** 常见的一个问题是如何让基于名字的虚拟主机使用**SSL**。答案是，目前还不能，正如第29章所介绍的那样。基于名字的虚拟主机依赖于HTTP请求的Host标头，但是，证书验证则是发生在**SSL**连接正在建立并且没有发送HTTP请求的时候。有一个协议用来把已有的HTTP升级为**TLS**，但是，大多数当前浏览器都不支持它（参见位于<http://www.rfc-editor.org/rfc/rfc2817.txt> 的RFC 2817）。

**Q:** 我可以和其他协议一起使用**SSL**吗？

**A:** `mod_ssl`模块把**SSL**协议实现为一个过滤器。使用同一个Apache服务器的其他协议可以很容易地利用**SSL**的优点。

## 30.8 实践练习

这个实践练习设计用来帮助你复习已经学过的知识。

## 问答题

1. 说出在Internet上执行安全通信所需的3个需求。
2. 如何启动一个支持SSL的Apache实例？

## 解答

1. 保密性、完整性和验证。
2. 确保httpd-ssl.conf文件通过指令包含在httpd.conf文件中，并执行apachectl start命令。

## 第31章 优化和调校MySQL

在本章，我们将学习：

- 对MySQL服务器的基本的硬件和软件优化技术。
- MySQL服务器的关键的启动参数。
- 如何使用**OPTIMIZE TABLE**命令。
- 如何使用**EXPLAIN**命令。
- 如何使用**FLUSH**命令清空表、缓存和日志文件。
- 如何使用**SHOW**命令获取有关数据库、表和索引的信息。
- 如何使用**SHOW**命令获取系统状态信息。

适当地关注和调整MySQL服务器可以使它正常地运行并且没有意外。系统的优化包括正确的硬件维护和软件调校。

### 提示：

要想了解维护和管理MySQL服务器的其他方法，考虑MySQL Workbench产品。可以在<http://www.mysql.com/products/workbench/> 找到这一功能丰富的图形界面的信息和截屏图。



## 31.1 构建一个优化的平台

设计一个结构良好、规范化的数据库架构，只是优化工作的一半（虽然是重要的一半）。另外一半就是构建和调校一个用来保存数据库的服务器。考虑一个服务器的4个主要部分：CPU、内存、硬盘驱动器和操作系统。这些部分中的每一个都需要提高速度，以及尽可能多的设计或编程来使数据库更快。

- **CPU** ——CPU越快，MySQL处理数据也就越快。这没有什么秘密，3.0GHz的处理器比1.0GHz处理器显然要快很多。随着处理器速度的稳定增长，并且随着价格越来越合理，数据库很容易提高速度。
- **内存** ——尽可能地多使用RAM。内存总是越多越好，而且RAM现在也很便宜。拥有足够的内存有利于缓解CPU的低速度。
- **硬盘驱动器** ——适当的硬盘驱动器不仅要足够大而且要足够快，才能满足数据库服务器及其流量。硬盘驱动器速度的一个重要指标就是寻道时间，或者说，驱动器转一圈并找到特定信息段所需的时间。寻道时间以毫秒为单位，并且对于桌面驱动器来说，平均寻道时间大约在8到9毫秒；对于服务器来说，平均寻道时间大约在3到5毫秒。当我们购买一个硬盘驱动器的时候，确保其大小足够容纳我们最终要存储在数据库中的数据，并且速度足够快速找到这些数据。
- **操作系统** ——如果我们使用的操作系统资源占用厉害（例如，Windows），那么有两种选择：购买足够的资源以供其使用，或者

使用那些不会耗尽资源的操作系统。

不管你是自己购买这些零部件来组装一台机器，还是购买定制服务器的管理解决方案，如果我们在系统级把正确的东西组合到一起，可能只需几步就能达到全面的服务器优化。

MySQL引擎的选择（MyISAM或InnoDB）也是一个优化选项。根据你的选择，各种附加的优化都可以使用。我推荐看一下MySQL手册中关于特定表的优化技巧，其地址是<http://dev.mysql.com/doc/refman/5.5/en/optimization.html>，还有位于<http://www.mysqlperformanceblog.com/>的MySQL Performance博客。

## 使用**benchmark()**函数

我们可以使用MySQL的**benchmark()**函数对服务器的速度进行一次快速测试，来看看处理一个给定表达式需要多长时间。我们可以使用比较简单的表达式，如10+10，也可以使用更为复杂一点的表达式，例如日期的提取。

不管表达式的结果是什么，**benchmark()**的结果总是0。**benchmark()**的目的不是得出表达式的结果，而是看看重复这个表达式一定的次数需要多长时间。例如，如下的命令执行10+10表达式100万次。

```
select benchmark(1000000,10+10);
```

这条命令在我的测试系统中的结果如下。

```

+-----+
| benchmark(1000000,10+10) |
+-----+
|                               0 |
+-----+
1 row in set (0.04sec)

```

下面这条命令把日期提取表达式也执行了100万次。

```
select benchmark(1000000, extract(year from now()));
```

这条命令在我的测试系统中的结果如下。

```

+-----+
| benchmark(1000000, extract(year from now())) |
+-----+
|                               0 |
+-----+
1 row in set (0.09sec)

```

重要的数字是用秒表示的时间数，也就是执行该函数所需的时间，第一个测试花了0.04秒，第二个测试花了0.09秒。我们可能要在一天中不同的时间多次同样地使用**benchmark()**（在服务器处于不同负载的情况下），从而对服务器的性能有一个更准确的了解。并且，尝试其他一些**benchmark**，它们会比这些简单示例给你的服务器带来更大的处理压力。

## 31.2 MySQL启动选项

MySQL AB提供了关于服务器性能参数的丰富信息，其中很多信息一般的用户肯定不需要使用到。坦白讲，如果你在虚拟托管环境中使用MySQL，除非请求修改服务器设置，否则你将无法使用这些信息。因此，不要完全被这些信息所包围，本节包含的只是可以更好地调校MySQL服务器的一些常用启动选项。

### 提示：

可以在位于<http://dev.mysql.com/doc/refman/5.5/en/server-system-variables.html>的MySQL手册阅读到更多的内容。当你启动MySQL的时候，一个名为my.cnf的配置文件载入。这个文件包含了从端口号码到缓存大小的信息，但是这些信息可以通过命令行启动选项来修改。

在MySQL安装目录的suppor-files子目录下或者在Windows的安装目录中，可以找到如下示例配置文件，每个都根据安装的内存的特定范围进行过调校。

- **my-small.cnf** ——对于RAM小于64MB的系统，其中的MySQL只是偶尔使用。
- **my-medium.cnf** ——对于RAM小于64MB的系统，其中MySQL是系统上的主要服务，或者对于系统升级到RAM 128MB，而MySQL和其他进程分享内存。这是最常见的配置，其中MySQL安装在作为一个Web服务器的同一个系统上，并且接受一个中度大小的流量。
- **my-large.cnf** ——对于RAM从128MB到512MB的系统，其中，MySQL是主要的服务。

- **my-huge.cnf** ——对于RAM从1GB到2GB的系统，其中，MySQL是主要的服务。

要使用这些文件的任何一个作为基本配置文件，只需要把选择的文件复制到/etc/my.cnf或my.cnf在你的系统上的位置，并且修改系统相关信息，例如端口或文件位置。

## 关键启动参数

有两个重要的启动参数，可能会对系统性能影响很大，这就是key\_buffer\_size和table\_cache。如果你只正确地调校了两个服务器参数，请确保是这两个。

key\_buffer\_size的值就是用于索引的缓存的大小。缓存越大，SQL命令完成和结果返回得越快。尝试在精细调校和过于优化之间找到更好的路线，我们可能在RAM为512MB的系统上拥有256MB的key\_buffer\_size，但任何超过256MB的设置可能导致系统性能的下降。

检查缓存的实际性能的一种简单方法，就是检查4个附加变量：key\_read\_requests、key\_reads、key\_write\_requests和key\_writes。我们可以通过执行SHOW STATUS命令来得到这些变量的值。

该命令将会返回一个长长的变量和值的列表，按照字母顺序排列。可以找到如下的一些行（你的值可能有所不同）。

Key_read_requests	10182771	
Key_reads	9326	
Key_write_requests	48487	
Key_writes	2287	

如果你用`key_reads`的值除以`key_read_requests`的值，结果应该小于0.01。另外，如果用`key_writes`的值除以`key_write_requests`的值，结果应该小于1。使用前面的值计算，我们得到的结果分别是0.000915861721998和0.047167281951863，正好处在可接受的参数范围内。我们可以尝试通过增加`key_buffer_size`的值来使这些值变得更小，但是，现在的值已经不错了。

另一个重要的服务器参数是`table_cache`，就是所有线程所打开的表的数目。默认值是64，但是，我们可能根据需要调整这个数值。使用`SHOW STATUS`命令，在输出中找到名为`open_tables`的变量。如果这个数值较大，那么`table_cache`的值应该增加。

包含在MySQL安装中的示例配置文件使用了`key_buffer_size`和`table_cache`的不同组合。我们可以使用这些组合作为基础来做出任何需要的修改。不管何时修改配置，都必须重新启动服务器以使修改生效。有时候，还不知道修改的结果是什么，在这种情况下，请确保在把修改提交到产品之前，在开发环境中尝试你的修改。

## 31.3 优化表结构

优化的表结构和设计良好的表有所不同。表结构优化必须在删除后重新利用未使用的空间，并且在做出结构修改之后对表进行基本清理。**OPTIMIZE TABLE**命令负责做到这些，它使用如下的语法。

```
OPTIMIZE TABLE table_name[,table_name]
```

例如，如果你想要优化testDB数据库中的**grocery\_inventory**表，使用**OPTIMIZE TABLE grocery\_inventory**。你可能会看到一条状态消息只是显示“OK”，或者看到一条消息显示“Table does not support optimize,doing recreate + analyze instead”。这两种情况都很好，因为其结果都是相同的，你的表已经优化了。

### 注意：

在进行优化的时候，这个表将被锁定，因此，如果表较大，应该在调度低谷或者当系统流量较小的时候执行优化。

## 31.4 优化你的查询

查询优化和索引的正确使用有很大关系。**EXPLAIN**命令检查一条给定的**SELECT**语句，看它是否达到了最佳的优化，是否已经在可能的地方使用索引。这对于包含**JOIN**的复杂查询特别有用。**EXPLAIN**的语法如下。

```
EXPLAIN SELECT statement
```

**EXPLAIN**命令的输出是包含如下信息列的一个表。

- **id** ——选择标识符ID。
- **select\_type** ——**SELECT**语句的类型，有多种类型。
- **table** ——表名。
- **type** ——连接类型，有多种类型。
- **possible\_keys** ——这个列表示MySQL将使用哪些索引来查找这个表中的行。如果结果是**NULL**，表示没有索引用来辅助查询。那么我们应该看看表结构，是否可以创建必要的索引来提高查询的性能。
- **key** ——查询中实际使用的键，如果没有使用索引，为**NULL**。
- **key\_len** ——使用的键的长度，如果有键的话。
- **ref** ——和键一起用来获取结果的任何列。
- **rows** ——为执行查询MySQL必须检查的行数。
- **extra** ——有关MySQL将如何执行查询的附加信息。有几个选项，例如**Using index**（使用了一个索引）和**Where**（使用了一条**WHERE**子句）。



对于“选择所有”的查询，并没有多少优化可做，除非添加一条使用主键的WHERE子句。possible\_keys列随后将显示PRIMARY，而Extra列将显示使用了Where。

当在包含JOIN的语句上使用EXPLAIN的时候，度量对查询的优化的一种快捷方法，就是查看rows列中的值。假设结果为2和1，把这些数字相乘，我们将得到2作为结果。这就是MySQL为了得到查询的结果而必须查看的行数。我们希望这个数字尽可能地低，而2已经相当低了。

要了解有关EXPLAIN命令的更多信息，请参阅位于<http://dev.mysql.com/doc/refman/5.0/en/explain.html> 的MySQL手册。

## 31.5 使用FLUSH命令

对一个特定的数据库具有重新加载权限的用户，可以使用FLUSH命令来清除MySQL所使用的内部缓存。通常，只有root级别的用户具有执行FLUSH这样的管理性命令的相应许可。

FLUSH的语法如下。

```
FLUSH flush_option
```

FLUSH命令具有9个不同的选项，最常用的选项如下。

- PRIVILEGES
- TABLES
- HOSTS
- LOGS

我们曾经在前面使用了FLUSH PRIVILEGES命令，在添加了新的用户之后使用。这条命令只是把MySQL数据库中授权的表重新载入，使得修改可以生效而不需要停止和重新启动MySQL。当我们执行一条FLUSH PRIVILEGES命令，得到Query OK响应确保清除过程顺利地进行了。

```
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.10 sec)
```

FLUSH TABLES命令关闭当前打开或使用的所有的表，并且在开始回到工作之前，基本上给MySQL服务器一毫秒的缓冲时间。当缓存为空，MySQL可以更好地使用可用内存。再一次，我们得到Query OK响

应。

```
mysql> FLUSH TABLES;  
Query OK, 0 rows affected (0.21 sec)
```

FLUSH HOSTS命令专门对主机缓存表起作用。如果我们无法连接到MySQL服务器，一个常见的原因是，对一个特定主机的连接达到了最大的连接数目并且会抛出错误。当MySQL看到很多的连接错误，它会假设出现问题并且直接阻塞试图到主机的任何其他连接。FLUSH HOSTS命令重新设置这一过程并且再次允许进行连接。

```
mysql> FLUSH HOSTS;  
Query OK, 0 rows affected (0.00 sec)
```

FLUSH LOGS命令关闭并重新打开所有的日志文件。如果日志文件变成一种负担，你希望开始一个新的日志文件，这条命令创建一个新的、空的日志文件。在一个文件中遍历一年的日志条目来查找错误，可能是一件繁琐的事情，因此，尝试至少每月清空日志。

```
mysql> FLUSH LOGS;  
Query OK, 0 rows affected (0.04 sec)
```

要了解有关FLUSH的更多信息，请参阅位于  
<http://dev.mysql.com/doc/refman/5.5/en/flush.html> 的MySQL手册。

## 31.6 使用SHOW命令

SHOW命令有几种不同的用法，它会产生输出，显示关于MySQL数据库、用户和表的众多有用信息。根据我们的访问级别，一些SHOW命令将不能供我们使用，或者只提供少量信息。root级别的用户具有使用所有SHOW命令的能力，而且能得到最全面的结果。SHOW的常见用法如下，我们稍后会详细介绍。

```
SHOW GRANTS FOR user
SHOW DATABASES [LIKE something]
SHOW [OPEN] TABLES [FROM database_name] [LIKE something]
SHOW CREATE TABLE table_name
SHOW [FULL] COLUMNS FROM table_name [FROM database_name] [LIKE something]
SHOW INDEX FROM table_name [FROM database_name]
SHOW TABLE STATUS [FROM db_name] [LIKE something]
SHOW STATUS [LIKE something]
SHOW VARIABLES [LIKE something]
```

SHOW GRANTS命令显示了一个给定用户在一个给定主机的权限。这是检查一个用户的当前状态的简单方法，尤其是如果我们拥有修改用户权限的请求的时候。通过SHOW GRANTS，我们首先检查看看该用户是不是已经拥有了所请求的权限。下面的例子查看用户joeuser的可用权限。

```
SHOW GRANTS FOR joeuser@localhost;
```

这个查询的结果如下。

```

+-----+
| Grants for joeuser@localhost |
+-----+
| GRANT ALL PRIVILEGES ON *.* TO 'joeuser'@'localhost' IDENTIFIED |
| BY PASSWORD ' *13883BDDBE566ECEFF0501CDE9B293303116521A' |
+-----+

1 rows in set (0.00 sec)

```

如果我们不是root级别用户或joeuser用户，将会得到一个错误。除非我们是root级别用户，否则，只能看到和自己相关的信息。例如，joeuser用户不允许查看有关root级别用户的信息。

```
SHOW GRANTS FOR root@localhost;
```

这条查询将导致如下的错误消息。

```
ERROR 1044: Access denied for user:'joeuser@localhost' to database 'mysql'
```

在本章剩余的部分，也请注意你的权限级别。如果你不是root级别用户，不可以使用其中一些命令，或者使用它们只能显示有限的信息。

还有一些其他常用的SHOW命令，要了解更多信息，请参阅位于<http://dev.mysql.com/doc/refman/5.5/en/show.html>的MySQL手册。

### 31.6.1 获取有关数据库和表的信息

在本书前面，我们已经使用了一些基本的SHOW命令来查看MySQL服务器上的数据库和表的列表。复习一下，SHOW DATABASE命令就是用来做这件事情的，它列出MySQL服务器上的所有数据库。如下是结果示例。

```

+-----+
| Database |
+-----+
| testDB   |
| mysql    |
+-----+
2 rows in set (0.00 sec)

```

当我们选择了要使用的数据库后，也可以使用SHOW列出数据库中的表。这个例子是在选择testDB数据库之后，运行一条SHOW TABLES查询的结果（你列出的表可能有所不同）。

```

+-----+
| Tables_in_testDB |
+-----+
| grocery_inventory |
| email             |
| master_name       |
| myTest            |
| testTable         |
+-----+
5 rows in set (0.01 sec)

```

如果为SHOW TABLES命令添加OPEN，将会得到表缓存中的所有表的一个列表，并显示它们被缓存和使用了多少次。

**SHOW OPEN TABLES;**

结果如下所示。

```

+-----+-----+-----+-----+
| Database | Table           | In_use | Name_locked |
+-----+-----+-----+-----+
| mysql    | procs_priv      | 0      | 0           |
| mysql    | db              | 0      | 0           |
| mysql    | host            | 0      | 0           |
| testdb   | grocery_inventory | 0      | 0           |
| mysql    | user            | 0      | 0           |
| mysql    | tables_priv     | 0      | 0           |
| mysql    | columns_priv    | 0      | 0           |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

把这些信息和本章前面所学习过的FLUSH TABLES命令结合起来使用，将能够帮助我们保证数据库更顺利地运行。如果SHOW OPEN TABLES显示表缓存了很多次，但是当前没有使用，使用FLUSH TABLES命令来释放内存。

### 31.6.2 获取表结构信息

一条有用的命令是SHOW CREATE TABLE，它所做的事情正如其名字所示，它显示用来创建指定的表的SQL语句。

```
SHOW CREATE TABLE grocery_inventory;
```

上述命令执行的结果如下。

```
+-----+-----+
| Table           | Create Table
+-----+-----+
| grocery_inventory | CREATE TABLE 'grocery_inventory' (
                        'id' int(11) NOT NULL auto_increment,
                        'item_name' varchar(50) NOT NULL default "",
                        'item_desc' text,
                        'item_price' float NOT NULL default '0',
                        'curr_qty' int(11) NOT NULL default '0',
                        PRIMARY KEY ('id')
                        ) ENGINE=InnoDB DEFAULT CHARSET=latin1
+-----+-----+
1 row in set (0.00 sec)
```

如果我们导出表结构，得到的信息和这个基本相同，但是，如果我们只是查找一个特定的表创建语句的提示或简单引用的话，SHOW CREATE TABLE命令使用起来会更快。

如果你需要知道表的结构但不需要SQL命令创建它，可以使用如下的SHOW COLUMNS命令。

```
SHOW COLUMNS FROM grocery_inventory;
```

查询的结果如下。

```
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null  | Key  | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| id         | int(11)       | NO    | PRI  |          | auto_increment |
| item_name  | varchar(50)   | NO    |      |          |                |
| item_desc  | text          | YES   |      |          |                |
| item_price | float         | NO    |      |          |                |
| curr_qty   | int(11)       | NO    |      |          |                |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

提示：

SHOW COLUMNS命令和DESCRIBE命令彼此互为别名，因此，它们做同样的事情。

SHOW INDEX命令显示了一个特定表中出现的所有索引的信息。  
语法如下。

```
SHOW INDEX FROM grocery_inventory;
```

这条命令产生一个信息完整的表，从列名到索引的基数。表31-1描述了这条命令所返回的列。

表31-1 SHOW INDEX结果中的列

列 名	说 明
Table	表的名字
Non_unique	1或0，1=索引可以重复，0=索引不可以重复
Key_name	索引的名字



Seq_in_index	索引的列序列号；从1开始
Column_name	列名
Collation	列的排列顺序，要么是A (升序)要么是NULL (不排序)
Cardinality	索引中唯一值的数目
Sub_part	在一个部分索引的列上，这里给出了索引字符的数目，如果整个键都是索引，则为NULL
Packed	数值列的大小
Null	显示列是否包含NULL值
Index_type	所使用的索引方法
Comment	任何附加的说明

产生一个填满结果的、广泛的表的另一条命令是SHOW TABLE STATUS命令。这条命令的语法如下所示。

```
SHOW TABLE STATUS [FROM database_name] LIKE 'something'
```

这条命令产生一个填满了信息的表，其范围从行的大小和数目到auto\_increment字段要使用的下一个值。表31-2描述了这条命令所返回的

各列。

表31-2      **SHOW TABLE STATUS**结果中的列

列      名	说      明
Name	表的名字
Engine	这个表所使用的存储引擎
Version	表的*.frm文件的版本
Row_format	行的存储格式：固定的、动态的或压缩的
Rows	行数
Avg_row_length	行的平均长度
Data_length	数据文件的长度
Max_data_length	数据文件的最大长度
Index_length	索引文件的长度
Data_free	分配了但没有使用的字节数
Auto_increment	一个auto_increment字段中使用的下一个值

Create_time	创建表的日期和时间（以日期时间格式）
Update_time	数据文件最后更新的日期和时间（以日期时间格式）
Check_time	表最后检查的日期和时间（以日期时间格式）
Collation	表的字符集和校对类型
Checksum	表的校验和值，如果使用校验和的话
Create_options	用于CREATE TABLE语句的任何附加选项
Comment	创建表的时候所添加的任何说明。此外，InnoDB表使用这个列来报告表空间中的空白空间

### 31.6.3 获取系统状态

SHOW STATUS和SHOW VARIABLES命令快速提供有关数据库服务器的重要信息。这些命令的语法就是SHOW STATUS和SHOW VARIABLES。

有超过300个状态变量作为SHOW STATUS的输出，但最有用的如下所示。

- **Aborted\_connects** ——尝试连接到MySQL的失败次数。任何时候，我们看到一个放弃的连接，就应该调查这个问题。可能和脚本中错误的用户名和密码有关，或者所允许的同时连接的数目设置得

太低，为了防止站点流量过大。

- **Connections** ——在当前正常运行时期，尝试连接到MySQL服务器的连接的总数。
- **Max\_used\_connections** ——在当前正常运行时期，同时使用的连接的最大数目。
- **Slow\_queries** ——超过long\_query\_time时间的查询的总数，long\_query\_time默认为10秒。如果较慢的查询多于一个，应该研究一下SQL语法。
- **Uptime** ——在当前正常运行时，服务器已经启动的总秒数。

在位于<http://dev.mysql.com/doc/refman/5.5/en/show-status.html> 的MySQL手册中，可以找到SHOW STATUS变量的一个完整列表，及其参数的说明。

SHOW VARIABLES命令产生325个结果，这些结果会控制MySQL的一般运行，并且包括如下项目。

- **connect\_timeout** ——显示MySQL服务器在一个连接尝试放弃之前等待的秒数。
- **max\_connections** ——在一个连接被拒绝之前，所允许到MySQL的同时连接的数目。
- **port** ——MySQL运行的端口。
- **table\_type** ——MySQL的表类型。
- **version** ——MySQL的版本号。

在位于<http://dev.mysql.com/doc/refman/5.5/en/show-variables.html> 的MySQL手册可以看到SHOW VARIABLES所返回的变量的一个完整列

表，以及它们的值的说明。当你知道了所拥有的值，就可以在MySQL配置文件或启动命令中修改它们。

## 31.7 小结

运行一个优化的MySQL服务器，首先从所使用的硬件和操作系统开始。系统的CPU应该足够快，并且应该有足够的RAM用来弥补CPU的不足。如果MySQL和其他的进程（如Web服务器）分享资源，尤其应该如此。

此外，所使用的硬盘驱动器也很重要，因为小的硬盘驱动器限制了可以在数据库中存储的信息的数量。硬盘驱动器的寻道时间也很重要，慢的寻道速度会导致服务器性能全面降低。操作系统应该不占用太多机器资源，并且应该和MySQL共享资源而不是自己使用所有的资源。

MySQL的一些关键启动参数包括`key_buffer_size`和`table_cache`，还有其他一些参数。可以在示例MySQL配置文件中找到基准值，或者可以修改这些变量并查看服务器性能，看看我们是否找到了适合环境的正确参数值。

除了硬件和软件优化，还有表的优化，以及SELECT查询的优化。表的优化使用OPTIMIZE命令，我们可以重新使用未使用的空间。我们可以通过使用EXPLAIN命令看看查询优化有多么好（或者多么不好）。最终的输出将会显示是否以及何时使用了索引，以及是否可以使用索引来提高给定查询的速度。

对MySQL服务器多加注意，以确保它持续顺利地运行。像FLUSH和SHOW这样的基本的管理命令，可以帮助我们识别和快速修复潜在的问题。所有这些命令都设计给MySQL1毫秒的喘息时间，如果它处在一

个很重的负载之下。众多的SHOW命令显示了有关数据库、表、索引，以及系统是如何运行的结构性信息。

## 31.8 Q&A

**Q:** 在单个的服务中，**MySQL**可以利用多**CPU**的优势吗？

**A:** 绝对可以。如果操作系统支持多CPU，MySQL将会利用其优点。然而，根据操作系统不同，使用多CPU的MySQL的调校有所不同。

**Q:** 要使用**OPTIMIZE**命令必须有什么级别的许可？

**A:** 对一个表具有INSERT权限的任何用户都可以执行OPTIMIZE命令。如果一个用户只有SELECT许可，则不能执行OPTIMIZE命令。



## 31.9 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 哪个MySQL函数使我们能够多次运行一个表达式以得出每次运行的速度？
2. 哪条SQL命令清除表的结构？
3. 哪条FLUSH命令重置MySQL日志文件？
4. 要快速确定MySQL是否支持InnoDB表，应该使用SHOW STATUS还是SHOW VARIABLES？
5. 编写一条SQL语句，使得我们能够看到用来创建myTable表的SQL语句。

## 解答

1. `benchmark()`函数
2. `OPTIMIZE`
3. `FLUSH LOGS`
4. `SHOW VARIABLES`
5. `SHOW CREATE TABLE myTable`

## 思考题

1. 如果你对服务器有root级别访问权限，修改key\_buffer\_size和table\_cache的值，并且在每次修改之后运行benchmark()函数，看看执行时间的区别。
2. 对数据库中所创建的所有表执行OPTIMIZE命令以清除任何结构问题。
3. 使用SHOW STATUS命令来获取有关MySQL服务器的信息，然后执行FLUSH命令清除服务器。在每条命令之后，再次使用SHOW STATUS来看看哪条命令影响到SHOW STATUS结果显示中的哪些结果。

## 第32章 软件升级

在本章，我们将学习：

- 如何跟踪最新的软件发布。
- 如何在**MySQL**的次版本之间升级。
- 如何在**Apache**的次版本之间升级。
- 如何在**PHP**的次版本之间升级。

整个本书中，我们已经提醒过你查找有关升级**PHP**、**Apache**和**MySQL**的新版本的信息，并且要留意更新。此外，还介绍了如何在构建的时候向**PHP**添加功能，但是这只是在安装软件的时候。在这个简短的一章中，我们将学习在经历了一段正常运行时间之后如何更新已经安装过的软件，而不会给系统带来严重破坏。

## 32.1 停留在循环中

你应该已经收藏了Apache、PHP和MySQL的站点。你曾经使用过这些技术6天还是6年都无关紧要，经常回头看看这些站点总是有必要的（我总是这么做）。如果访问Web站点的主要原因是获取有关更新的信息，你可以订阅一个自愿订阅的邮件列表。

- 要了解MySQL信息，到<http://lists.mysql.com/> 并订阅MySQL信息列表。
- 要了解Apache信息，到<http://www.apache.org/foundation/maillinglists.html> 并订阅Apache新闻和信息列表。
- 要了解PHP信息，到<http://www.php.net/mailling.lists.php> 并订阅PHP信息列表。

### 何时升级

正如安装章节所提到的，任何时候，当开发者发现有必要的时候就会发布次版本变化的新软件，而没有任何特定时间表。但是，仅仅次版本发生了变化，并不一定意味着你应该马上跟进并升级软件。然而，有时候还是应该升级。

软件发布通常遵守major.minor.revision的格式，例如PHP 5.4.0的主版本是5，次版本是4，修订编号是0。也有可能是这样，实际上修订中所做的修改可能“很小”，但是发布还是还是将其当做一次修订。

当公布一个安全性修正的时候，应该立即升级软件。通常，安全性问题直到暴露的时候才会被发现，有时候是在测试环境，但有时候是因为一个仅仅想要引发麻烦的恶意用户而发现的。在证实了一个安全性问题之后，你可以确定它已经成为需要开发者修复的最优先的问题，并且很快将会看到一个升级的发布。这时候，你应该立即升级，即便你没有使用引发安全问题的特定元素。漏洞就是漏洞，为什么让它隐藏呢？

下面是Apache 更新日志的一个例子，它记录了版本2.0.61和2.0.63之间发生的变化（版本2.0.62没有公开发布），这是升级所需要的一个指路牌。

```
SECURITY: CVE-2007-6388 (cve.mitre.org)
mod_status: Ensure refresh parameter is numeric to prevent a possible XSS
attack caused by redirecting to other URLs.
```

首要的原则是，如果单词“安全性（security）”出现在更新日志的任何位置，那么应该立即升级。在PHP5 更新日志中（位于<http://www.php.net/ChangeLog-5.php>），将与安全相关的修改收集到一起，并且由于其重要性而放在这个列表的顶部。

然而，如果只是一个维护发布，意味着它包含了在正常的开发中发生的错误修复和一般性维护，你可能不需要立即丢弃原有内容来升级软件。下面是Apache和PHP更新日志中维护性项目的一些例子。

```
mpm_winnt: Eliminate wait_for_many_objects. Allows the clean shutdown of the
server when the MaxClients is higher then 257, in a more responsive manner.
Fixed bug #43137 (rmdir() and rename() do not clear statcache).
```

如果更新列表中没有和你、你的工作以及你的环境相关的东西，那么，你可以推迟更新，直到日程安排中的休息日或者一个雨天。例如，如果在PHP的一个维护发布中修复的所有bug都是和Windows平台相关

的，而你是在Linux上运行PHP，那么你可以将其搁置一边，高枕无忧。

即便我们没有立即更新软件，至少紧跟在软件的当前产品版本之前的一个或两个次版本，这是一个好主意。只有通过这样，才更可能添加新的特性或得到真正和我们的工作或环境相关的漏洞修复。



## 32.2 升级MySQL

无论你是使用UNIX/Linux，还是使用Windows，升级MySQL的次版本都很简单，只要安装新的版本就好像其他版本不存在一样。

提示：

在升级到MySQL的一个新的次版本之前，备份好已有的数据库。

在同样的基本版本中更新MySQL很容易，5.4是基本版本，5.5是一个基本版本，以此类推。这意味着在5.5.x组内的任何次版本升级就像在旧软件之上安装新软件一样的简单。通过Windows Installer，你可以看到这一切。从二进制发布安装的UNIX/Linux用户，只需要把“mysql”符号目录重新链接到新的、未解包的发布，以此作为安装进程的一部分。

如果在升级过程中遇到问题，参考位于<http://dev.mysql.com/doc/refman/5.5/en/upgrading.html> 的故障排除说明。然而，升级MySQL的次版本对我来说总是顺利的过程，在哪个平台上都是如此。

如果在升级的时候修改了字符集，你必须运行如下的命令调整已有数据的校对类型。

```
# myisamchk -r -q --set-collation=new_collation_name
```

## 32.3 升级 Apache

和MySQL一样，升级或重新编译Apache和第一次安装软件遵从同样的步骤。Windows用户得益于Installer应用程序，它会自动检测之前的版本，删除核心部件并安装新的。然而，Windows Installer将会保留已有的配置文件。我们还要负责升级任何其他的特定于版本的模块，例如mod\_ssl，该模块是绑定到Apache的特定版本的。

对于UNIX/Linux用户，这个过程遵照和最初安装同样的步骤。当我们解压缩新的发布包，它创建一个以新版本号命名的目录。例如，如果我们之前的版本号是2.2.17，并且要升级到2.4.1，对应的目录名将会分别是httpd-2.2.17和httpd-2.4.1。

Apache的实际安装目录由你确定，当你运行configure脚本的时候确定，如下面的例子所示。

```
# ./configure --prefix=/usr/local/apache2
```

在运行了configure脚本生成了Apache的新版本之后，只要像第一次安装Apache那样经历make和make install过程。

现在，你应该想要在旧的Apache版本上直接安装Apache新版本了，甚至可以在旧的httpd二进制仍在运行的时候这么做。只要确保备份了配置文件以防出错。然而，如果你更习惯在不同的目录中安装新版本，这也不错，只是必须要把所有和Web相关的文件（即文档根目录下的所有内容）移动到新目录下，并且让所有相应的编辑都反映到新的httpd.conf文件。选择什么方法取决于你，一种方法需要较多的文件移动，另一种

方法需要较多的配置。

在UNIX/Linux上升级了Apache之后，应该也重新编译PHP模块。Windows用户不需要有一个重新编译的模块，但是应该确保相应的与PHP相关的更新也出现在httpd.conf文件中，这关系到驻留在PHP目录树下的模块的加载。

## 修改Apache而不需要升级

假设我们需要从Apache添加或删除功能而不需要升级到一个新的次版本。一个例子就是，添加一个新的模块，或者把系统上使用的OpenSSL升级到新的版本。

在这种情况下，UNIX/Linux用户应该找到已有的源目录（例如httpd-2.2.17），并且在命令行输入**make clean**。基本上，这将重置makefiles以便我们可以重新编译Apache而不需要依赖以前的、缓存的值。在make clean命令之后，用新的参数运行configure脚本，并且再次进行make和make install过程。在这种情况下，应该不需要重新编译PHP模块。

通过在httpd.conf中对相应的行去掉注释，或者如果这些行已经不存在就添加那些行，Windows用户能够激活预编译模块。

## 32.4 升级PHP

既然UNIX/Linux用户可以通过各种编译选项来添加如此多的功能到PHP，你升级或修改PHP可能比Apache和MySQL要频繁很多。不管你是升级到一个新的次版本，还是简单地添加新功能，或者删除某些不再需要的功能，修改已有版本的过程确实和第一次安装它的过程相同：`configure`、`make`和`make install`。`make install`步骤把PHP模块放入到Apache目录树的相应位置。当新的模块放入到指定位置，重新启动Apache，新版本的PHP就开始使用了。

如果你升级到PHP的一个新的次版本，当我们提取了发布文档，将会生成一个根据版本号的完全不同的目录树。在新的目录结构中执行`configure`、`make`和`make install`步骤，并且将会生成一个新的PHP模块，它独立于其他的模块。

Windows用户有一组不同的任务要执行，向一个已有的模块添加新功能只需要通过在`php.ini`中去掉其条目的注释并重新启动Apache，就可以激活该模块。升级到一个新的次版本则需要下载一个新的发布文件。这个文件的内容随后提取到一个按照它所表示的版本而命名的目录中。然后，必须按照安装所需的步骤，相应地配置`php.ini`，因为每个版本生成一个不同的文件。最后，在Apache的`httpd.conf`文件中修改和PHP相关的路径名，并且重新启动服务器，新的PHP版本就可以使用了。

### 使用PECL和PEAR扩展PHP

可以从PECL（PHP Extension community library，PHP扩展公用库，

位于<http://pecl.php.net/> ) 和PEAR (PHP Extension and application repository, PHP扩展与应用库, 位于<http://pear.php.net/> ) 获取用户创建的扩展和应用程序。这些站点都有规则和样式管理, 因此, 从这里下载的任何内容都具有很高的质量。

如果你在查找PHP安装的其他扩展, 请查看PECL。如果你在查找集成到应用程序中的开源代码的库, 请查看PEAR。

## 32.5 小结

这个简短的一章提供了升级MySQL、Apache和PHP当前版本安装的一些指导。我们学习了如何找到更新，以及如何评估升级到新版本的重要性。此外，还学习了升级或修改MySQL、Apache和PHP的步骤。

## 32.6 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 对于主版本为3，次版本为4，修订号为14的软件，我们该如何表示它。 ^
2. 升级到任何软件的一个新的次版本所考虑的首要原因是什么？
3. 什么命令将清空之前的makefiles和缓存设置？



## 解答

1. 完整的版本号应该是3.4.14。
2. 开发者发现或修复了安全性问题。
3. `make clean`命令。

## 思考题

在阅读到本章的时候，可能已经有了与本书CD所包含的或者你在本书开始所下载安装的PHP、MySQL和Apache的不同版本了。因此，如果你能够升级（也就是说，如果你没有使用虚拟主机环境的话），选择一种或多种方法，并完成升级过程。

## 第33章 使用应用程序框架

在这个简单的一章中，我们将学习如下内容：

- 应用程序框架是什么，它们有什么用。
- 软件架构的模型-视图-控制器模式的相关基础知识。
- 考虑和安装一些流行的**PHP**应用程序框架。

本书已经教授了在一个Web站点中创建动态功能的基础知识，既包括使用单个的脚本来增强功能或显示动态数据，也包括将一系列的脚本以某种一致的形式结合在一起，以生成基于Web的应用程序。当你试图开始一个较大的项目的时候，你很可能想要使用其他人很可能已经在自己的项目中用到过的功能，而你没有时间也不愿意重新发明轮子，这时候，应用程序框架会成为你的新的好朋友。

## 33.1 理解应用程序框架

究其本质，应用程序框架就是一组库和模板，它们允许你快速地开发功能丰富的动态站点和Web应用程序，而不需要从头开始构建每一个模块。你可能还记得，在本书第五部分中，我总是很仔细地提到，给出的示例只是创建脚本实现总体目标的众多方法中的一种。使用应用程序框架，使得你可以说“我知道有很多种方法来创建一个登录过程（或购物车、论坛等等），并且，这些方法不是从头开始，可以用应用程序框架的方法来实现。”

除了针对常用功能重用一个稳定的代码这一明显的好处，使用框架还可以帮助开发者保持一致的软件架构模式。在PHP框架的例子中，这种模式通常是模型-视图-控制器（MVC）模式，我们将在下一小节中更详细地介绍它。框架以及由此引发的某些软件架构所具有的另一个方面优点是，能够为应用程序实现一种稳定且一致的三层架构。在一个三层架构中，我们有一个表示客户端的物理层，以及一个在你的控制管理下的应用程序和数据库。换句话说，用户的Web浏览器或者移动设备（客户端）直接连接到应用程序（通常是应用程序的表现层，或者说你在浏览器中所看到的内容），并进而进入数据库以获取要显示的数据。

了解一些内容管理系统是很重要的，它们都是可以自行安装的软件包，例如WordPress (<http://www.wordpress.org>)、Drupal (<http://www.drupal.org>)和Joomla (<http://www.joomla.org>)。从技术上讲这些软件包都不是应用程序框架，但是，可以安装和定制它们以创建你自己喜欢的风格，从而重用它们的功能丰富的代码。

## 33.2 使用MVC模式

任何软件架构模式的目标之一是，提供一致的结构、元素以及特性和属性等，以确保应用程序内部的工作对于那些开发者和维护者来说是透明的。考虑一下这种情况，一个人独自编写了一款应用程序，而没有遵从任何结构性的模式。当一个新人加入团队，或者最初的编写者离开的时候，将会发生什么事情？没有一个基础的和共享的知识体系，例如，遵从一种通用的架构模式，那么，新人需要花费大量的时间才能熟悉起来，更不要说他能够解决进行审计、测试和维护乱糟糟的代码所碰到的问题了。

MVC软件架构模式是为基于Web的应用程序而准备的，并且实际上，很多应用程序（或者甚至只是动态站点）都遵从这一模式的某个版本，而这么做毫不费力。图33-1以一种基本的方式说明了MVC的概念，其中的实线表示三个组件中的每两个之间的直接关联，而虚线表示某些部分之间有时候会有间接的关联。

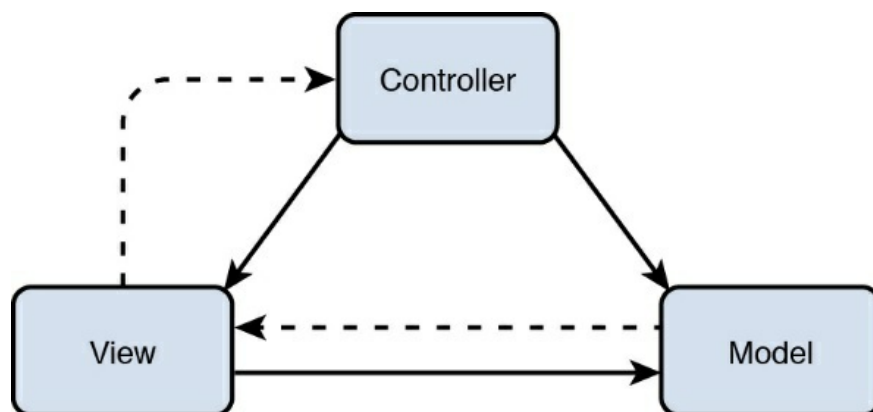


图33-1 MVC模式的图形表示

通常，MVC软件架构模式中的3个组件可以定义如下。

- 模型 ——存储并分隔数据。它本身不是一个数据库，但是，可以将其看作是脚本中确定好访问数据库并从中提取信息的部分，然后，它会将提取的信息显示给用户。
- 视图 ——模型中的数据可视化表示。这就像你的这样一部分脚本，它们确定不做任何其他事情，只是显示你从数据库获取的数据。
- 控制器 ——系统中的一个操作被调用的方式。通常，我们称之为应用程序的“大脑”，它确定如何使用模型来得到操作的一个成功的结果。

还有其他的一些方法来思考一个MVC应用程序的流程。假设网上商店的一位用户正在浏览商品页面，并且想要将商品添加到她的购物车中。

1. 用户使用鼠标按下页面上的一个提交按钮。
2. 控制器接收表单操作并且将信息发送给模型。
3. 为了向用户显示一个操作并更改其已变化的数据，视图和模型交流数据，以将数据取出并显示给用户。
4. 控制器等待更多的用户操作，然后，这个循环将重复。

本章中所介绍的每个PHP框架都使你可以很容易地将MVC模式应用到你的软件应用程序。很多其他的PHP框架也这么做，尽管你可能不会选择该模式，但还是推荐你这么做，以使得应用程序后续的测试、开

发、部署和维护更容易。

要了解关于MVC模式的更多示例和说明，请参见Jeff Atwood的博客文章“Understanding Model-View-Controller”，位于<http://www.codinghorror.com/blog/2008/05/understanding-model-view-controller.html>。

### 33.3 安装和使用PHP应用程序框架

在编写本书的时候，全球的开发者使用超过30种的PHP应用程序框架。即使选择常用的一些来介绍的话，工作量也不小。我在本章中挑选的这些有着（相对）较长的历史，并且其开发者社区有着上升的趋势。实际上，在你评估一款框架是否适用于自己的用途的时候，有3个和代码不相关的方面需要考虑：它是否有一段时间了，以及是否稳定？人们积极地使用它吗？开发它的公司或开发者组织积极地维护它吗？

选择应用程序框架的时候的其他考虑还包括如下几个方面。

- 确定框架是否最适合于你要开发的应用程序的类型；一些框架很适合于电子商务，一些框架很适合于内容发布，有些框架则两者都适合。
- 确定该框架是否为你提供了机会以使用一种软件架构模式，如果是这样的话，它是否是你想要使用的模式。
- 确定该框架是否需要额外的PHP模块或服务器库。如果是，但是无法控制自己的服务器，并且由此无法修改安装的库和模块的话，你将无法使用该框架。

接下来的几个小节介绍3种流行的开源PHP应用程序框架，Zend、CakePHP和CodeIgniter。再次强调一下，实际上不止这3个框架可用；Symfony (<http://www.symfony.com/>)和Yii([http:// www.yiiframework.com/](http://www.yiiframework.com/))也经常进入“最佳候选”的清单。Wikipedia维护了PHP应用程序框架的一个全面的列表，位于



[http://en.wikipedia.org/wiki/Comparison\\_of\\_web\\_application\\_frameworks#P](http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks#P)

。

### 33.3.1 Zend Framework

如果你要开发企业级的PHP应用程序的话，应该认真地考虑从Zend Framework开始(<http://framework.zend.com/>)。你可能听说过Zend公司，也就是开发Zend Framework的公司，其创始人从PHP语言诞生的时候就是该语言的贡献者之一。该框架核心的PHP引擎常常称作Zend引擎。换句话说，如果按照我前面提到的那些标准来评估Zend Framework的话，很难找到一个比它更加稳定、时间更长或者说更多人积极参与开发的框架了。

尽管该软件的名字叫做Zend Framework，并且实际上它也是框架，你也许还听到过人们称之为组件库，因为我们也可能从一组松散耦合的组件中挑选部分使用，而不是要实现一个结构化的、架构良好的应用程序。例如，Zend的单独组件分别解决了数据库连接和存档、国际化和本地化、认证、授权、会话管理、Web服务的使用和暴露、邮件功能等等诸多问题。

这个框架的各个组件根本没有任何负面的评价，它确实保证了让我们时刻记得使用框架的目标之一，就是坚持一种易于维护的架构模式，你的应用程序将会很整齐一致。但如果你的应用程序包含了框架组件的一个库，并且你只是使用其中的一种，你可能要考虑一下维护框架的过度负担（更新、打补丁等等）是否值得。

你的答案可能是有所不同的，但是你应该知道什么样的决定更适合

自己。要下载和安装Zend Framework以使其与已安装的PHP、Apache和MySQL一起工作，首先访问位于<http://framework.zend.com/manual/en/requirements.introduction.html> 的 Requirements 页面，以确保你拥有PHP的正确版本或者任何附加的库和模块。然后，访问<http://framework.zend.com/download/latest>，找到该软件的完整版或精简版的链接。在获取了你所选取的格式的发布文件后，只需要根据发布版中的说明来提取它就可以了（简而言之，解压缩一个目录并将其放入到已有的文件系统中）。Zend Framework 快捷入门指南位于<http://framework.zend.com/manual/en/learning.quickstart.intro.html>，通过示例应用程序的创建，向你展示了使用该框架及其组件的整个过程。

### 33.3.2 CakePHP

另一个长时间存在的稳定的PHP应用程序框架是 CakePHP(<http://www.cakephp.org/>)，它也有一个强大的用户和开发者社区。实际上，CakePHP是一个MVC框架，其组件具备数据库连接、验证、授权、会话管理、Web服务使用和发布等常见功能，这些都和Zend Framework相似，实际上，和很多其他的框架也是相似的。

CakePHP最大的一个卖点是易于使用（或者说易于集成），而其简单性并不影响到开发和企业部署的丰富功能。和Zend Framework一样，你不必处理定制的配置文件的，只需要下载和安装框架并开始快速开发。从新用户的角度来看，CakePHP的优势在于除了具备标准的开发者论坛之外，它还具有详细的和用户友好的文档，以及多媒体的教程。

要下载CakePHP以配合已有的PHP、Apache和MySQL使用，首先访

问CakePHP Cookbook的安装页面

（<http://book.cakephp.org/2.0/en/installation.html>），以确保你的PHP和任何附加的库和模块的版本正确。然后，从链接进入下载页面或者站点的主页，下载当前的版本。

获取了发布文件以后，只需要把它解压缩到你的Web服务器的文档根目录。在开发环境和产品环境中使用CakePHP的时候，你有几个不同的选项，详细介绍参阅Cookbook（也可以访问位于<http://book.cakephp.org/2.0/en/getting-started.html>的快捷入门指南，以了解使用CakePHP创建一些示例应用程序的完整过程。

### 33.3.3 CodeIgniter

不仅CakePHP号称易于使用，CodeIgniter(<http://www.codeigniter.com>)也是初学者的一个很好的开端，其功能面向发布内容，而不是商务过程或其他的事务流程。这并不是什么坏事情，因为在架构全面的软件中以一种易于使用的方式来发布内容，这是锦上添花的事情。我可能不会在企业中使用CodeIgniter，但很多Web开发者由此可以很好地让众多的客户感到满意，而不需要进入企业空间。因此，再次强调一下，要选择适合你的框架。

CodeIgniter是一个MVC框架，此外，它还提供了诸如数据库连接、验证、授权、会话管理、本地化、图像操作、使用和发布Web服务等常见的功能，很像是本章中讨论的其他的框架。和本章所介绍的其他框架一样，其使用过程包括下载和安装框架，然后就可以立即用来进行开发。和CakePHP一样，CodeIgniter也胜在除了具备标准的开发者论坛之外，还具有详细的和用户友好的文档，以及多媒体的教程。

要下载CodeIgnite以配合已有的PHP、Apache和MySQL一起使用，首先访问位于[http://codeigniter.com/user\\_guide/general/requirements.html](http://codeigniter.com/user_guide/general/requirements.html)的Server Requirements页面，以确保PHP和任何附加的库和模块的版本正确。然后从位于<http://codeigniter.com/downloads/>的下载页面下载正确的版本。

## 33.4 小结

本章介绍了使用应用程序框架的概念，并且特别介绍了使用模型-视图-控制器模式的软件架构。然后，介绍了如何评估你自己的应用开发所要使用的框架。最后，介绍了3种常用的PHP应用程序框架（Zend、CakePHP和CodeIgniter）。

## 33.5 实践练习

这个实践练习设计用来帮助你预测可能的问题、复习已经学过的知识，并且开始把知识用于实践。

## 问答题

1. 使用应用程序框架有哪些优点？
2. 在MVC模式中，模型做些什么？
3. 你必须使用应用程序框架吗？

## 解答

1. 使用一个稳定的代码，遵从一种软件架构模式，并且不用重新发明轮子。
2. 模型存储并从控制和视图组件分隔数据。
3. 并非如此。实际上，本书前面章节所介绍的PHP和MySQL开发的基本方面，并不依赖于任何框架。



## 思考题

1. 下载并安装至少一种本章中所介绍的框架（如果你安装了多种框架，删除旧的框架以避免冲突）。
2. 阅读至少一种这些框架的开发者所提供的教程，以便获得使用框架和MVC模式的一些实用知识。

# 欢迎来到异步社区！

## 异步社区的来历

异步社区([www.epubit.com.cn](http://www.epubit.com.cn))是人民邮电出版社旗下IT专业图书旗舰社区，于2015年8月上线运营。

异步社区依托于人民邮电出版社20余年的IT专业优质出版资源和编辑策划团队，打造传统出版与电子出版和自出版结合、纸质书与电子书结合、传统印刷与POD按需印刷结合的出版平台，提供最新技术资讯，为作者和读者打造交流互动的平台。

# 新年新气象

社区UI全新改版，崭新面貌迎接2017！为答谢社区用户，

即日起  
1月26号

全场电子书8折优惠！



前端开发



数据科学



编程语言



移动开发



游戏开发

## 机器学习&深度学习

更多>>



Python机器学习——预测分析核心算法



贝叶斯方法：概率编程与贝叶斯推断



机器学习项目开发实战



贝叶斯思维：统计建模的Python学习法

免费电子书  
Free eBook

立即领取

我要写书  
Write for Us

立即查看

近期活动

## 社区里都有什么？

### 购买图书

我们出版的图书涵盖主流IT技术，在编程语言、Web技术、数据科学等领域有众多经典畅销图书。社区现已上线图书1000余种，电子书400多种，部分新书实现纸书、电子书同步出版。我们还会定期发布新书书讯。

### 下载资源

社区内提供随书附赠的资源，如书中的案例或程序源代码。

另外，社区还提供了大量的免费电子书，只要注册成为社区用户就可以免费下载。

### 与作译者互动

很多图书的作译者已经入驻社区，您可以关注他们，咨询技术问题；可以阅读不断更新的技术文章，听作译者和编辑畅聊好书背后有趣的故事；还可以参与社区的作者访谈栏目，向您关注的作者提出采访题目。

## 灵活优惠的购书

您可以方便地下单购买纸质图书或电子图书，纸质图书直接从人民邮电出版社书库发货，电子书提供多种阅读格式。

对于重磅新书，社区提供预售和新书首发服务，用户可以第一时间买到心仪的新书。

用户帐户中的积分可以用于购书优惠。100积分=1元，购买图书时，在  里填入可使用的积分数值，即可扣减相应金额。

### 特别优惠

购买本电子书的读者专享异步社区优惠券。使用方法：注册成为社区用户，在下单购书时输入“57AWG”，然后点击“使用优惠码”，即可享受电子书8折优惠（本优惠券只可使用一次）。

## 纸电图书组合购买

社区独家提供纸质图书和电子书组合购买方式，价格优惠，一次购买，多种阅读选择。



## Wireshark网络分析的艺术

作者：林沛满

责编：傅道坤

分类：计算机科学 > 安全与加密 > 网络安全

Wireshark是当前最流行的网络包分析工具。它上手简单，无需培训就可入门。很多棘手的网络问题遇到Wireshark都能迎刃而解。

本书挑选的网络包来自真实场景，经典且接地气。讲解时采用了生活化的

[更多>>](#)

[下载PDF样章](#)

[配套文件下载](#)

5.6K  
浏览

57  
想读

7  
推荐

分享：[微信](#) [QQ](#)

纸质 ¥45.00-¥31.50 (7折)

电子 ¥25.00

电子 + 纸质 ¥45.00

[购买](#)



+



总价：75.60

[一起购买](#)

(纸质)

(纸质)

[目录](#)

[评论 9](#)

[勘误 1](#)

[出版信息](#)

作者简介 [00](#)

专业书评 [00](#)

内容提要 [00](#)

### 本书作译者



LinPeiman

上海

1.0K经验值

[发私信](#)

[送积分](#)

[关注](#)

《Wireshark网络分析就这么简单》即《Wireshark网络分析的艺术》作者

### 兑换样书

[立即兑换](#)

[如何赚取积分](#)

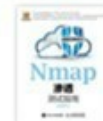
### 电子书版本

[PDF](#)

[Epub](#)

[Mobi](#)

### 精彩推荐



Nmap渗透测试指南

作者：商广明

## 社区里还可以做什么？

### 提交勘误

您可以在图书页面下方提交勘误，每条勘误被确认后可以获得100积分。热心勘误的读者还有机会参与书稿的审校和翻译工作。

### 写作

社区提供基于Markdown的写作环境，喜欢写作的您可以在此一试身手，在社区里分享您的技术心得和读书体会，更可以体验自出版的乐趣，轻松实现出版的梦想。

如果成为社区认证作译者，还可以享受异步社区提供的作者专享特色服务。

### 会议活动早知道

您可以掌握IT圈的技术会议资讯，更有机会免费获赠大会门票。

## 加入异步

扫描任意二维码都能找到我们：



异步社区





微信订阅号



微信服务号



官方微博



QQ群: 436746675

社区网址: [www.epubit.com.cn](http://www.epubit.com.cn)

官方微信: 异步社区

官方微博: @人邮异步社区, @人民邮电出版社-信息技术分社

投稿&咨询: [contact@epubit.com.cn](mailto:contact@epubit.com.cn)